أكاديمية المبرمجين العرب

الإكليل (الجزء الأول)

الشامل في أساسيات البرمجة بلغة الفيجوال بيسك



www.mobarmijoun.com

"يا أيها الذين آمنوا اتقوا اللّه و قولوا قولا سديدا. يصلح لكم أعمالكم و يغفر لكم ذنوبكم ومن يطع اللّه و رسوله فقد فاز فوزا عظيما"

الأحزاب : 70 و 71

الفهرس العام:

مقدمة الحقيبة البرمجية:

بسم الله الرحمن الرحيم، الحمد لله رب العالمين والصلاة والسلام على الرحمة المهداة للعالمين، سيدنا محمد وعلى آله وصحبه أجمعين، وعلى التابعين، ومن تبعهم بإحسان إلى يوم الدين ثم أما بعد:

بات لائحا لكل مهتم بالشأن المعلوماتي، أن المحتوى البرمجي بالعربية نادر جدا قياسا مع أقرانه في اللغات الأجنبية، بل ويفتقر في كثير منه إلى الأساليب البيداغوجية لتوصيل المعلومة بالشكل الذي ينبغي أن تصل عليه به إلى القارىء الهدف، فكان المحتوى البرمجي العربي عبارة عن شظايا معلومات متفرقة لا تلبي الغرض ولا تحقق المبتغى.

في الجانب المكتوب، نجد عطاء يشكر باذلوه عليه إلا أنه يفتقد إلى التنظيم والترتيب وفي أحايين كثيرة إلى الإكمال والإتمام. وفي الجانب المرئي يتفاقم الوضع ليجد المتعلم نفسه أمام دورات تستهويه بأسمائها لكنها لا تشبع نهمه المعرفي بمحتوياتها، إذ أن موادها لا تحوي جديدا ولا تغطي مفاهيم يتعطش إليها، وخير دورة يحسن صائغوها تقديم محتواها يجدها المتعلم مجرد اجترار لمعلومات مستهلكة منتشرة على منابر معرفية عديدة.

فلما كان الأمر كذلك تطلعنا بشغف في أكاديمية المبرمجين العرب إلى لم الشعث وسد الثغر بشيء من الجهود التعليمية، فنشرنا قبل زهاء خمس سنوات مجموعة من الكتب العربية ضمن سلسلة "كن أسدا" والتي لاقت رواجا واسعا واستقبلها المتعلمون بالقبول الحسن، ثم أتبعنا ذلك بمجموعة من الدروس المرئية كان أبرزها "دورة سي

شارب الكاملة" و"دورة إنشاء برنامج إدارة المبيعات" حيث عرفتا انتشارا واسعا في صفوف الطلبة والمعاهد والمبرمجين الأحرار مما حفزنا وبعث فينا روح الأمل والإيجاب فقررنا مواصلة المسير واختياض لغات برمجية أخرى واقعة ضمن تخصصاتنا.

ولأننا بفضل اللّه قد زاولنا لغة الفيجوال بيسك دوت نيت مدة زمنية ليست يسيرة واشتغلنا على مختلف إصدارات الدوت نيت فريموورك بدء من الإصدار 1.0 الذي صاحب نسخة الفيجوال ستوديو 2003، قررنا أن نجنح نحو هذه اللغة حتى نوفيها حقها من الكتابة والتسجيل المرئي لعل اللّه جل وعلا يفتح على أيدينا عقولا شغفتها لغة الفيجوال بيسك حبا فلم يجدوا لولعهم بها من يلبيه.

في هذا الإطار ومن هذا المنطلق، جاءت فكرة الحقيبة البرمجية باقتراح من أخينا الفاضل "حسين العبدلي" من بلاد الحرمين، الذي رأى لي رأيا غير الذي كنت أنتويه فكان لي خير موجه وأفضل مرشد، إذ رأى أنه من الحكمة والتعقل أن تكون الدروس متسلسلة وفق أجزاء لكل جزء طابعه الخاص، وأن يصاحب كل جزء من هذه الأجزاء كتاب يعود المتعلم إليه في حال استعصى عليه مفهوم في التسجيلات المرئية أو العكس. وإحقاقا منا لبعض أفضال هذا الأخ الغالي على قلوبنا أدعو كل منتفع بهذه الحقيبة البرمجية أن يخصه بدعوة في ظهر الغيب وأن يتولاه المولى تبارك وتعالى بحفظه هو وسائر أهله الكرام.

حول الكتاب:

في هذا الكتاب، الذي يعد الجزء الأول من سلسلة كتب الحقيبة البرمجية للغة فيجوال بيسك 2015، سوف نطلع على أساسيات البرمجة بأسلوب تدريجي، ينطلق بنا من التعرف على جهاز الحاسوب باعتباره الجهاز الذي يشتغل عليه المبرمجون، وفهم أهم مكوناته المادية المتدخلة في عمليات تخزين ومعالجة البيانات، مستعرضين في غضون ذلك مفاهيم: البرنامج Program، البرمجة Programming، لغات البرمجة Programming Languages: أقسامها وأهميتها في صياغة الحلول التقنية وصناعة البرمجيات.

ويحتوي الكتاب أيضا على شرح مستفاض لجميع المفاهيم الأساسية في البرمجة، من قبيل المتغيرات Variables وأنواع البيانات Data Types والروابط Operators والبنيات الشرطية Conditions والحلقات Loops والمصفوفات Arrays وإدارة الاستثناء ات Exceptions Handling، ومعالجة النصوص والعملات والتواريخ String, Currency and Date Handling وغير ذلك من المفاهيم المهمة بلغة الفيجوال بسيك دوت نيت 2015.

ويشمل الكتاب كذلك أمثلة تطبيقية لكل مفهوم برمجي مرتبة على شكل خطوات، يسبقها شرح نظري سلس من أجل تهيئة الأذهان وتحضيرها لفهم دور هذا المفهوم في العملية البرمجية واستيعاب آليات استخدامه، مما يخول للقارىء أن يلم بكافة جوانبه النظرية والعملية والبيداغوجية.

بعد هذا الجزء من الكتاب، يأتي الجزء الثاني الخاص بشرح نمط البرمجة الكائنية التوجه Object Oriented Programming، والذي سنتناول فيه بحول الله مختلف المفاهيم المنضوية تحت هذا الأسلوب البرمجي.

أسأل الله العلي القدير أن أكون قد وفقت في تصنيف هذا الكتاب، و أن يجعله خالصا لوجهه الكريم، مع متمنياتي لكم بالتوفيق والسداد ودام لكم البشر والفرح !

خالد السعداني

بنية الحاسوب

تعريف وجيز لجهازالحاسوب / الحاسب

الحاسوب هو جهاز الكتروني مثله مثل باقي الأجهزة الالكترونية (تلفاز، هاتف، جهاز تسجيل،...) يستخدم لتخزين ومعالجة البيانات، وهو يتكون من جزءين لا ثالث لهما، أحدهما آلي Hardware وهو الجانب المادي الذي يضم مكونات الحاسوب التي نراها ونلمسها، أما الجزء الثاني فهو الجزء البرمجي Software وهو الجانب الخفي المسؤول عن تشغيل البرامج والألعاب والملتيميديا من خلال تحكمه في الجانب الآلي Hardware.

مكونات الحاسوب المادية

يتكون الحاسوب من مجموعة من العناصر المادية نذكرها فيما يلي أهمها:

الذاكرة الرئيسية أو الحية RAM(Random Access Memory):

يمكننا تعريف الذاكرة بأنها مجموعة من الخانات المتتالية والمرقمة عبر عناوين، وكل خانة يمكنها أن تحتوي على بيانات، تتم معالجتها من قبل وحدة المعالجة، كما يمكن للذاكرة أن تقوم بتخزين البرامج (البرنامج هو مجموعة من الأوامر المتسلسلة التي يتم تنفيذها للحصول على نتيجة معينة)، ويتم تمثيل البيانات في الذاكرة على شكل ثنائي عبر متتاليات من الأصفار والآحاد كما سنرى فيما بعد.

كل خانة في الذاكرة مرقمة لكي يسهل الوصول إلى محتواها من قبل وحدة المعالجة، ويسمى هذا الترقيم بالعنونة Addressing، أي أن كل خانة لها عنوانها الخاص Address.

ويمكننا تمثيل الذاكرة الرئيسية بهذا الشكل:



الشكل 1 التمثيل الاصطلاحي للذاكرة الرئيسية

وحدة معالجة البيانات Central Processing Unit:

وهو الجزء المهم في الحاسوب، ويعد بمثابة الدماغ المسؤول عن تنفيذ كل عمليات معالجة البيانات المخزنة في الذاكرة.

ويقوم بكل العمليات الحسابية (الجمع، الطرح، الضرب، القسمة) ويقوم أيضا بالعمليات المنطقية مثل مقارنة البيانات.

تقوم وحدة المعالجة بأخذ الأوامر المخزنة في الذاكرة على شكل بيانات، وتبدأ في تنفيذها بدء من أول أمر وانتهاء بآخر أمر وتقوم بإجراء العمليات الحسابية والمنطقية

الواردة في البرنامج المخزن، وكلما اقتضى الأمر تقوم بتخزين الناتج في الذاكرة لتستعمله مع أوامر أخرى، وفي ختام تنفيذ البرنامج تقوم وحدة المعالجة بإرسال النتيجة إلى الجهاز الخاص بعرضها (مثلا طباعة نتيجة عملية حسابية في نافذة الكونسول، ستقوم وحدة المعالجة بإرسال النتيجة إلى الشاشة)

الأجهزة Devices:

وهي كل الأجهزة الموصولة بالحاسوب وهناك من يقسمها إلى:

أجهزة الإدخال Input devices : لوحة المفاتيح، سكانر، قارىء الأقراص،...إلخ.

وأجهزة إخراج Output Devices: الشاشة، الطابعة، مكبرات الصوت،...إلخ.

وأجهزة التخزين Storage Devices: أقراص صلبة، مفاتيح اليو أس بي، الأقراص، الديسكيت، ...إلخ.

اللغة التي يفهمها الحاسوب

رغم عظم المهام والعمليات التي يقوم بها الحاسوب، إلا أنه ليس ذو ذكاء خارق قياسا مع العقل البشري، فهو لا يفهم سوى رقمين 0 و 1 (وهذا تقدير اصطلاحي فقط لا علاقة له بما يتم فيزيائيا على مستوى الحاسوب)، كل البيانات سواء كانت عبارة عن فيديو، صورة، صوت، أو أي ملف آخر، فإن الحاسوب لا يراها سوى على شكل سلاسل من

الأصفار والآحاد المخزنة في الذاكرة والتي تتم معالجتها وقراءتها بواسطة وحدة المعالجة، لكي تترجم إلى الشكل الذي نراها عليه.

أصفار وآحاد هي لغة الحاسوب، ولأنهما إثنان "2" سميت هذه اللغة ب "اللغة الثنائية Binary Language"، رغما أن هذه التسمية في المعلوميات الغرض الوحيد منها هو تبيان أن الحاسوب يفهم قيمتين متعارضتين فقط، وتم استخدام الأرقام 0 و 1 دلالة على ذلك، والأصل أن المسألة الكترونية، تنبني على التيار المتدفق، والمعلومات هي تسلسل لحالة التيار، لذلك تجد البعض يمثل اللغة الثنائية بالعبارة التالية "تيار يمر، تيار لا يمر" كناية على طبيعة البيانات التي يفهمها الحاسوب. هذا ما يقع فيزيائيا، لكن لفهم هذه المسائل تقنيا، يتم استخدام الثنائي 0 و 1 لتمثيل البيانات.

لمحة

تمثيل البيانات التي يفهمها الحاسوب عبر الترميز الثنائي **هو** تمثيل **تجريدي** واصطلاحي يدل على قيمتين حقيقيتين التي يفهمها الحاسوب بصفته جهازا الكترونيا (دارة مفتوحة، دارة مغلقة)، ونستخدم التمثيل الثنائي استنادا على لوحدتين 0 و 1 **ويسمى** بالترميز الثنائي Binary encoding، على غرار الترميز العشري Decimal encoding الذي يستخدمه الإنسان (الأعداد العشرة المعروفة من 0 إلى 9) لتمثيل المعلومات، وبعيدا عن الفيزياء يمكننا القول: **بأن الحاسوب لا يفهم سوى الأصفار والآحاد لتمثيل البيانات.**

تاريخ لغة الفيجوال بسيك

قبل أن نخوض في بحر الدوت نيت، دعونا نستهل درسنا بالتعرف على أساسيات وأصول البرمجة لكي نكون على بينة من أمرها.

البرمجة تعني كتابة أوامر بإحدى لغات البرمجة وتوجيهها إلى الحاسوب لكي يقوم بتنفيذها من أجل أداء عمل ما، وتسمى هذه الأوامر التي نكتبها للحاسوب برنامجا Program، وتسمى اللغة التي نكتب بها هذه الأوامر لغة برمجية Programming Language.

توجد العديد من لغات البرمجية مثل سي بلس بلس، جافا، سي شارب، بايتون، فيجول بزيك دوت نيت ولغات أخرى.

الغرض من استخدام لغات البرمجة هو اقتسام المشقة بين الإنسان والحاسوب، لأن الحاسوب لا يفهم لغاتنا البشرية، ولأننا نحن البشر لا نفهم لغة الحاسوب التي تكنى بلغة الآلة Machine Language وكذلك Machine Code أو اللغة الثنائية Binary Language، وسميت كذلك لأنها تستند في تمثيل البيانات ومعالجتها على وحدتين ثنتين وهما الصفر والواحد.

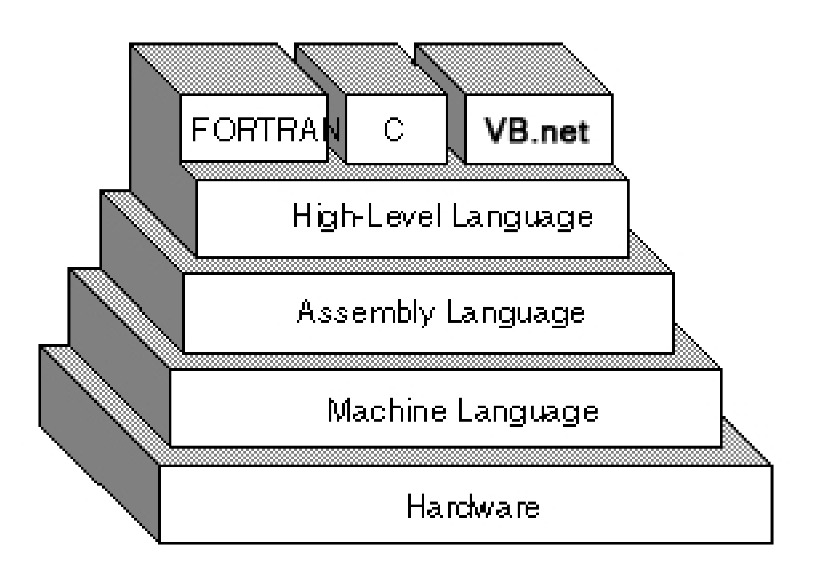
فلكي يكون الحوار بين المبرمج والحاسوب متناغما يتم استخدام لغات وسيطة تسمى لغات برمجية من أجل كتابة أوامر مفهومة نسبيا عند الانسان ومن السهل على الحاسوب أن يعالجها ويحولها إلى أصفار وآحاد من خلال عمليات الترجمة التي تنفذ بواسطة برنامج يسمى المترجم Compiler وفي بعض اللغات بواسطة المفسر Interepter.

تعتبر لغة الفيجوال بزيك دوت نيت والتي سننطقها اختصارا VB.Net، من أبرز وأقوى اللغات البرمجية، وهي من إنتاج شركة ميكروسوفت الشهيرة، وتأتي لغة VB.Net ضمن إطار العمل الذي يسمى بالفريموورك دوت نيت والذي يدعم لغات أخرى إضافة إلى VB.Net مثل سي شارب C#، وسي بلس بلس C++، وأف شارب F# ...

وقد تتفاجأ أخي الفاضل إذا علمت بأن تاريخ لغة البزيك Basic التي تنتمي إليها لغة VB.Net يعود إلى عام 1963، أي عمرها الآن أكثر من خمسين سنة، وتمكننا لغة VB.Net من إنتاج برامج ويندوز أي برامج وتطبيقات تشتغل على بيئة الويندوز، وتمكننا كذلك من إنتاج برامج مرتبطة بقواعد البيانات، وتمكننا كذلك من إنشاء مواقع وتطبيقات ويب ديناميكية قوية ومتقدمة عبر استخدام تقنية ASP.Net، وأنواع أخرى من المشاريع سنمر عليها إن شاء الله في مختلف أجزاء هذه الحقيبة البرمجية.

لغة VB.Net هي لغة عالية المستوى High-Level Programming Language أي أن صيغة كتابة الأوامر بها بعيدة جدا عن لغة الآلة وقريبة أكثر من اللغة الانجليزية.

وتعرض لنا الصورة التالية تصورا شكليا لمفهوم قرب وبعد لغات البرمجة من لغة الآلة:



الشكل 2 أنواع اللغات البرمجية

كلما كانت اللغات البرمجية قريبة من لغة الآلة سميت باللغات المنخفضة المستوى Low Level Languages وكمثال على ذلك شاهد لغة الأسمبلي أعلاه، وكلما ابتعدت اللغات البرمجية عن لغة الآلة واقتربت من لغة الإنسان سميت باللغات العالية المستوى High Level Languages وكمثال عليها شاهد لغات الباسكال والفيجوال بيسك دوت نيت...

ذكرنا أن لغة VB.Net تنتمي إلى عائلة لغات البزيك التي يعود تاريخ ميلادها إلى عام 1963، لكن النسخ الأولى من لغتنا ظهرت في عام 1990 مع إصدار لغة Visual Basic 1.0 التي استمرت ميكروسوفت في إنتاجها وإصدار نسخ منها إلى غاية النسخة Visual Basic 6، لتبدأ بعد ذلك في الأفول والسقوط، وبمقابل ذلك ظهر الفيجوال بزيك دوت نيت وبدأ يكتسب شهرة كبيرة بين صفوف المبرمجين بسبب سهولته وسلاسته المستنبطة من عائلة البزيك، وبسبب قوته التي يستمدها من نطاق الفريموورك.

بالنسبة للبرنامج الذي يمكننا من كتابة برامج بلغة VB.Net فهو يسمى Microsoft Visual Studio، وتوجد به إصدارات كثيرة آخرها هو النسخة 2015 والتي سنستخدمها خلال حصصنا إن شاء الله.

الفيجوال ستوديو

الفيجوال ستوديو هو برنامج من إنتاج شركة ميكروسوفت، يمكننا من إنشاء البرامج بالعديد من اللغات البرمجية بما فيها لغة الفيجوال بزيك دوت نيت، ويمكننا من إنشاء أنواع مشاريع مختلفة من تطبيقات ويندوز وتطبيقات ويب وتطبيقات موبايل وغيرها...

آخر نسخة تم إصدارها من هذا البرنامج إلى حدود تأليف هذا الكتاب هي النسخة 2015 والتي سنستخدمها في حقيبتنا البرمجية هذه إن شاء الله.

ويمكننا تحميله من الرابط التالي:

http://www.microsoft.com/en-us/download/details.aspx?id=44934

وهذا رابط مباشر يسمح لك بتحميل البرنامج:

http://download.microsoft.com/download/4/A/0/4A0D63BC-0F59-45E3-A0FF-9019285B3BC5/vs2015.preview_ult_ENU.iso

وبالنسبة لكيفية تحميله وتثبيته، يمكنك العودة إلى الدرسين الخامس والسادس.

الدوت نيت فريمووورك

الدوت نيت فريمووورك هو إطار عمل أنشأته شركة ميكروسوفت لإتاحة تطوير البرامج بأكثر من لغة برمجية، وهو عبارة عن طبقة فاصلة بين نظام التشغيل وبين البرامج التي ننشؤها بإحدى لغاته، حيث يوفر لنا كافة المكتبات والفئات التي نحتاجها.

مكتبة الفئات (Framework Class Library)

وهي مجموعة من الفئات التي يمكننا استخدامها في برامجنا التي ننشؤها بإحدى لغات الدوت نيت، وتمكننا من بناء أنواع مشاريع مختلفة، على سبيل المثال: الفئات Windows Forms هي فئات تسهل علينا عملية إنشاء برامج بواجهات استخدام Graphical User Interfaces (GUI)، كذلك الفئات Web Forms تمكننا من بناء تطبيقات ويب بتقنية asp.net.

كل مجالات الأسماء والفئات التي يمكننا استخدامها في برامجنا التي نصممها بلغات الدوت نيت موجودة داخل هذه الباقة المسماة مكتبة الفئات Framework Class Library.

ماهي MSIL و CLR؟

MSIL هي اختصار ل Microsoft Intermediate Language وتعرف كذلك بالاختصار IL الذي يعني Intermediate Language وكذلك CIL اختصارا ل Common Intermediate Language وهي الصيغة التي يتم تحويل شفرات الدوت نيت إليها بعد عملية الترجمة Compiling، هذه اللغة الوسيطة تحول بدورها إلى لغة الآلة في زمن التنفيذ Runtime أو بعد تنصيب البرنامج على الحاسوب بواسطة مترجم مشهور بالاسم JIT وهي اختصار ل Just-In-Time Compiler.

جميع برامج الدوت نيت يتم تنفيذها على آلة افتراضية تسمى CLR اختصارا ل Common Language Runtime وهي البرنامج المسؤول عن تنفيذ البرامج المكتوبة في بيئة الدوت نيت وتضمن CLR للبرامج الأمور التالية:

آلية التنظيف التلقائي Automatic Garbage Collector:

وتعني أن CLR تقوم بإدارة الذاكرة عبر تحريرها من جميع الكائنات Objects التي انتهى دورها ولم تعد تؤشر إلى نوع ما، بمعنى أنك بصفتك مبرمجا لست مطالبا بتفريغ الذاكرة وتحريرها من الموارد التي لم تعد مستخدمة لأن CLR تعفيك من هذه المهمة عبر آلية Garbage Collector.

حماية وصول الكود Code Access Security:

تعرف اختصارا ب CAS، وتعني هذه الخاصية أن CLR تأخذ بعين الاعتبار صلاحيات النظام الذي يتم تنفيذ الكود عليه، بمعنى لو أن الكود يحتوي على أمر ما من

شأنه التعديل أو حذف ملف محمي بصلاحيات معينة فإن CLR تدير هذا الأمر وتحول بين الكود وبين الوصول إلى عناصر الحماية الخاصة بالجهاز الذي ينفذ الكود عليه.

التحقق من الكود Code Verification:

هذه الخاصية تعني أن CLR تهتم بضمان سلامة تنفيذ الكود كأن تمنع البرنامج من حجز مكان في الذاكرة غير مسموح بحجزه، وكذلك إدارة الاستثناءات والأخطاء الواردة في البرنامج Handling Exceptions.

تحويل الكود الوسيط إلى لغة الآلة:

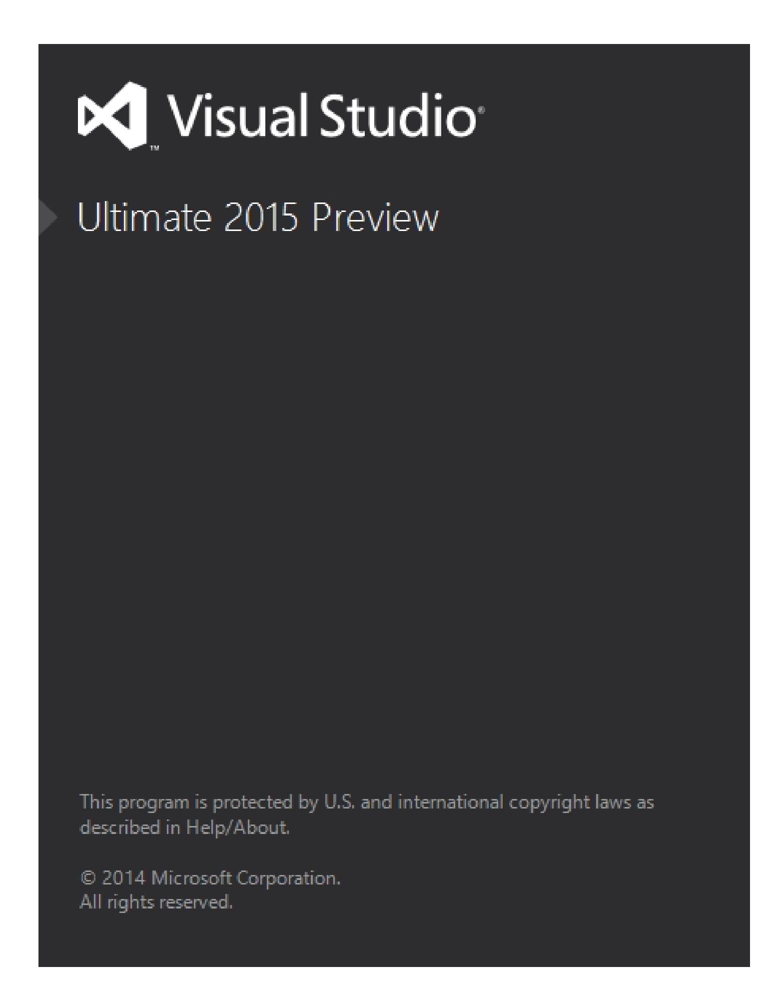
تستخدم CLR المترجم JIT من أجل تحويل الكود IL إلى لغة الآلة لتقوم بعد ذلك بتنفيذه.

وتشكل CLR إضافة إلى مكتبة الفئات إطار العمل دوت نيت فريموورك.

دوت نيت فريموورك = CLR + مكتبة الفئات (Framework Class Library)

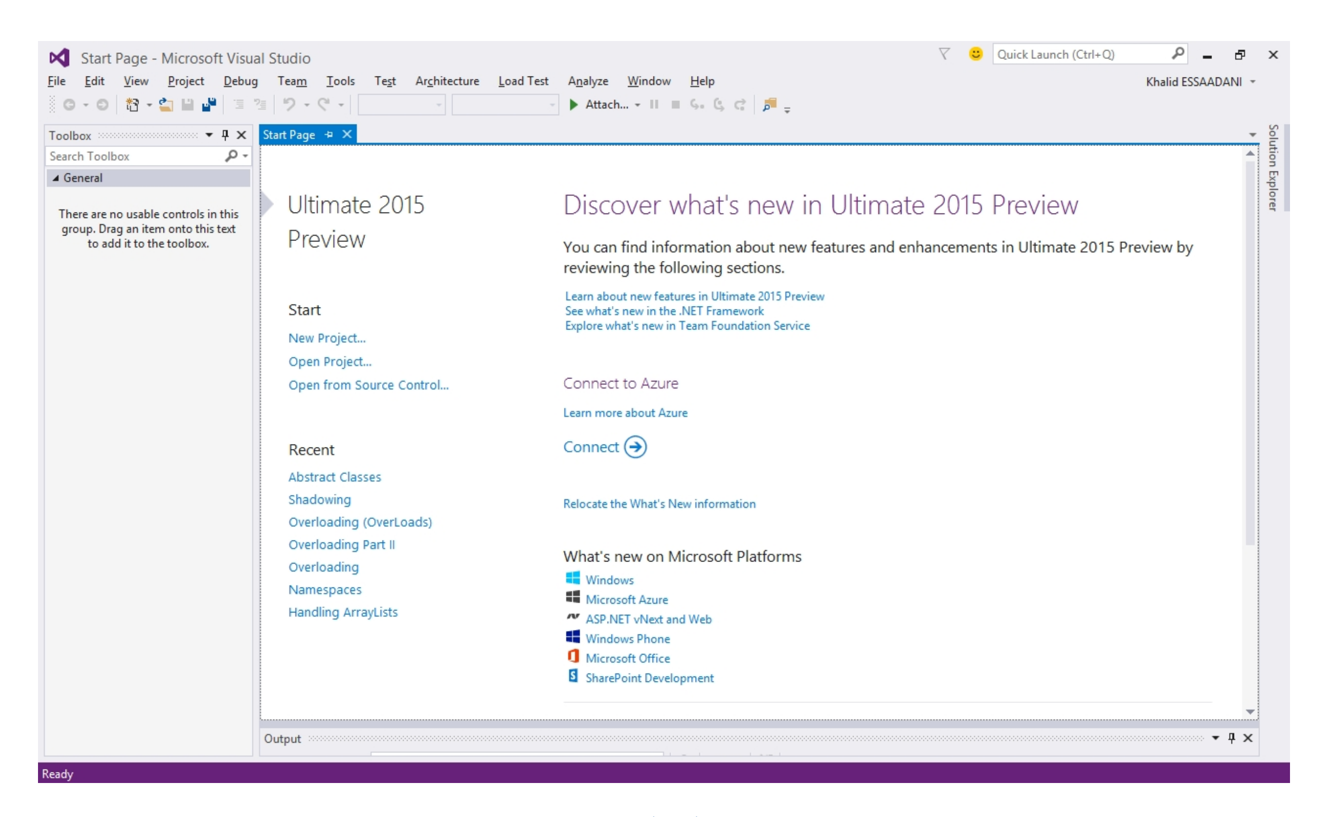
إنشاء مشروع جديد:

نقوم أولا بفتح برنامج ميكروسوفت فيجوال ستوديو عبر الذهاب إلى القائمة ابدأ Start في الويندوز، ثم نكتب في خانة البحث Visual Studio ويظهر لنا البرنامج في نتائج البحث، فنقوم بالضغط عليه، لتطالعنا شاشة البدء الآتية:



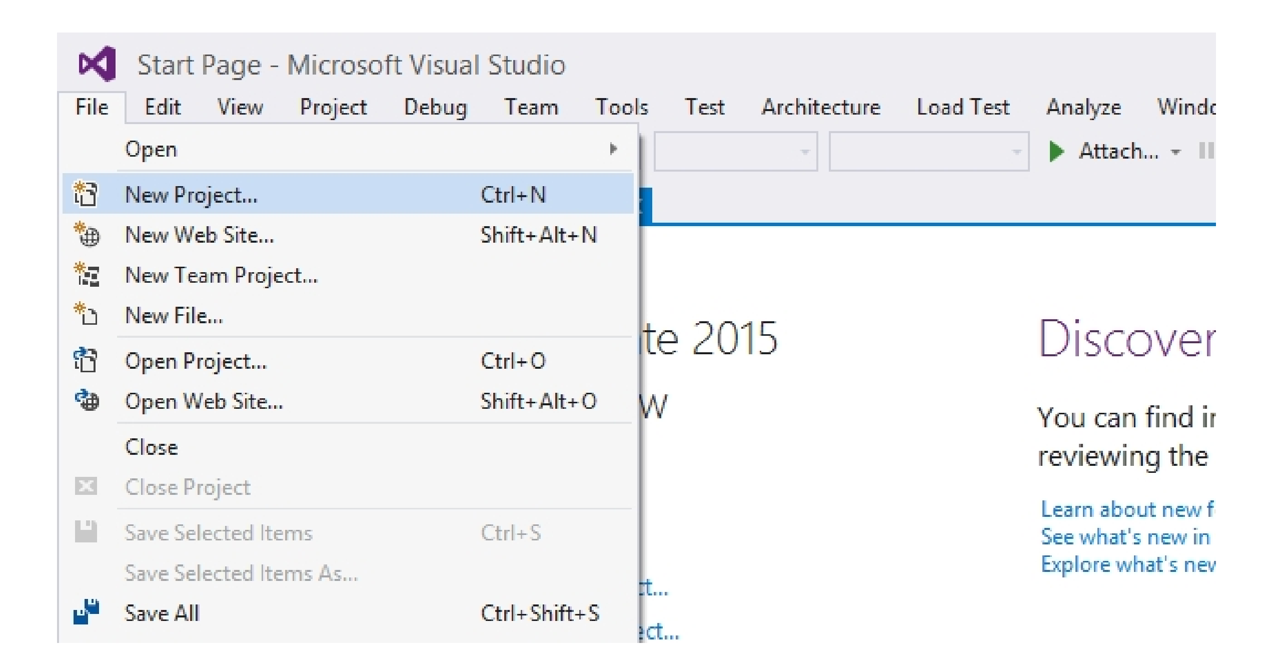
الشكل 3 صورة انتظار الفيجوال ستوديو

ننتظر قليلا حتى تظهر لنا شاشة الانطلاق في الفيجوال ستوديو وهي كما يلي:



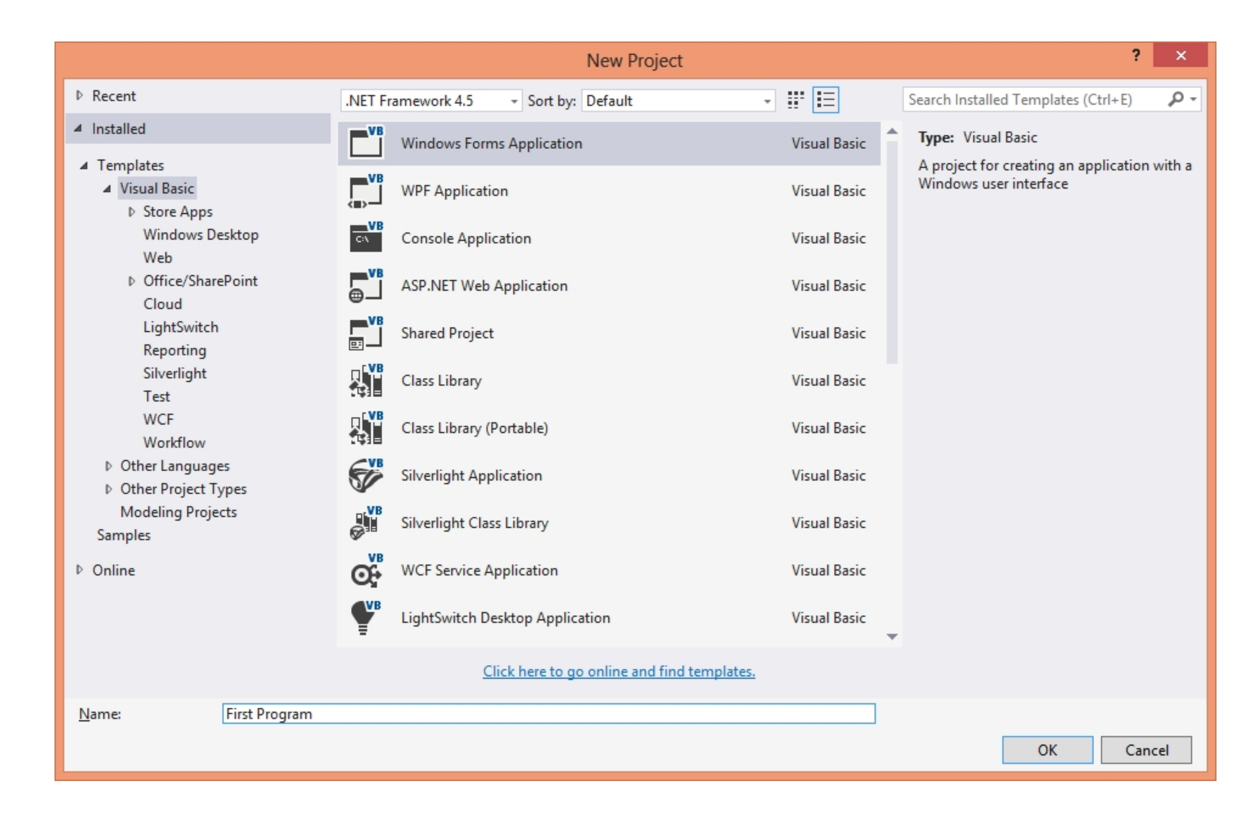
الشكل 4 شاشة بداية الفيجوال ستوديو

لإنشاء مشروع جديد، نذهب إلى القائمة File ثم نختار الأمر مشروع جديد New Project كما تعرض الصورة الآتية:



الشكل 5 إنشاء مشروع جديد

لتطالعنا الشاشة التالية:



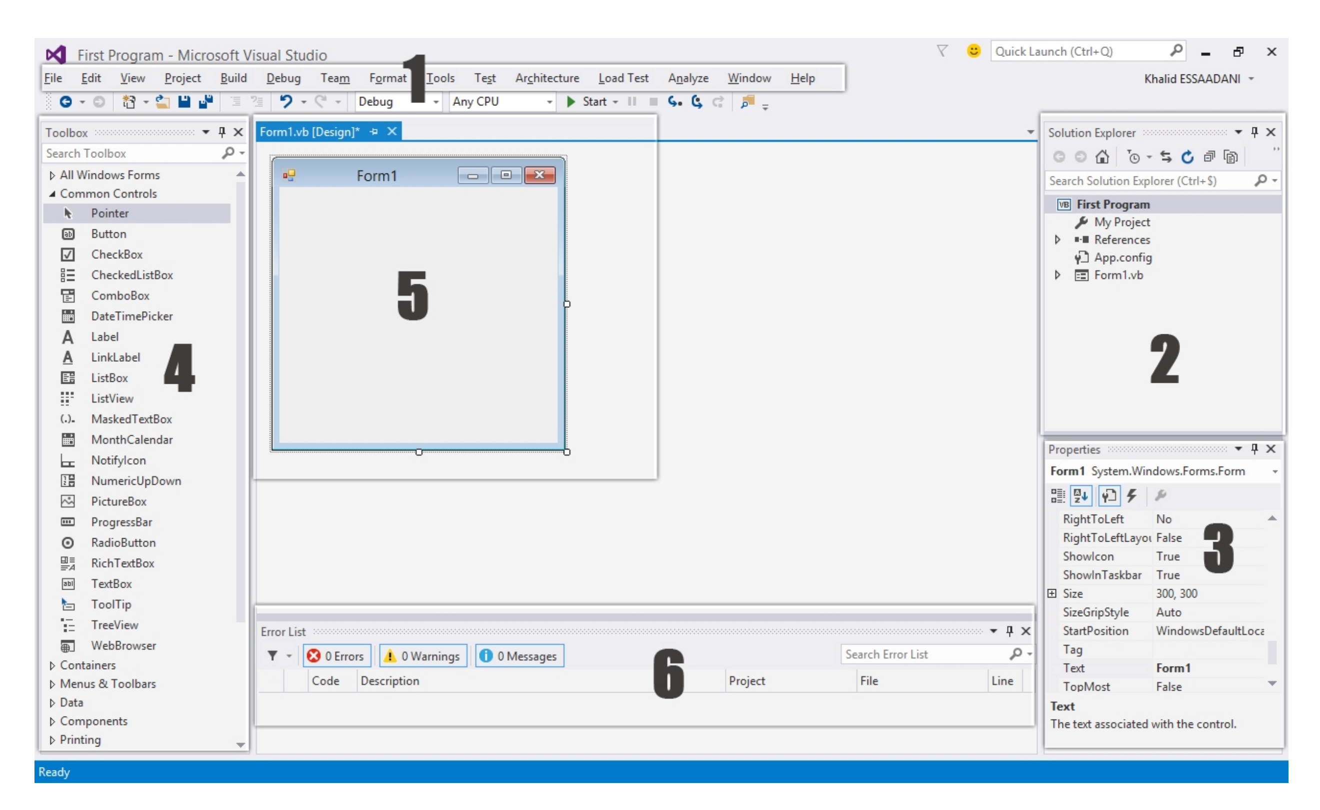
الشكل 6 اختيار خصائص المشروع **(لغة البرمجة ونوع المشروع)**

الشاشة أعلاه تمكننا من اختيار لغة البرمجة التي نريد استخدامها في المشروع، وتمكننا كذلك من تحديد نوع المشروع المراد إنشاؤه (مثلا برنامج من نوع ويندوز، أو من نوع ويب...)

نقوم باختيار لغة البرمجة فيجوال بيسك Visual Basic، ثم نحدد نوع المشروع Windows Forms Application ونقوم بإدخال اسم المشروع وليكن مثلا First Program.

بعد ذلك نضغط على الزر OK وننتظر قليلا ريثما يتم تهيئة بيئة التطوير وفتح المشروع الجديد الذي أنشأناه للتو.

بيئة التطوير لمشاريع الويندوز



الشكل 7 بيئة التصميم لمشاريع الويندوز

تعرض الصورة أعلاه أهم مكونات بيئة التصميم الخاصة بمشاريع الويندوز وهي كالتالي حسب الأرقام:

1. شريط القوائم:

ويحتوي على كافة القوائم التي نحتاجها في مشاريعنا مجمعة حسب نوع مهامها، مثلا الأوامر التي تسمح بإنشاء مشروع جديد، أو حفظ الملف الحالي أو ملفات المشروع أو فتح مشروع سابق كلها موجودة داخل القائمة File، والأوامر التي تسمح بالنسخ واللصق والتراجع والحذف وغير ذلك موجودة في القائمة Edit، والأوامر الخاصة بالمشروع من قبيل إضافة ملفات جديدة إليه أو استيراد مكتبات وغيرها موجودة في التبويب

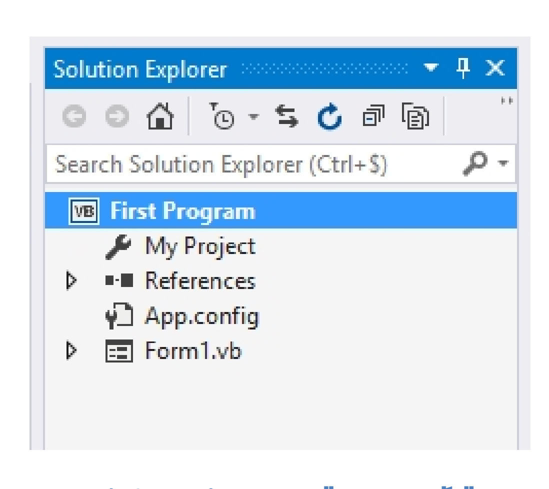
Project، وهكذا دواليك مع باقي القوائم، كل واحدة تحتوي علي الأوامر التي تشترك في طبيعة المهام، وهذه صورة لمختلف القوائم التي يتيحها لنا برنامج الفيجوال ستوديو:



الشكل 8 شريط القوائم

2. متصفح المشروع Solution Explorer:

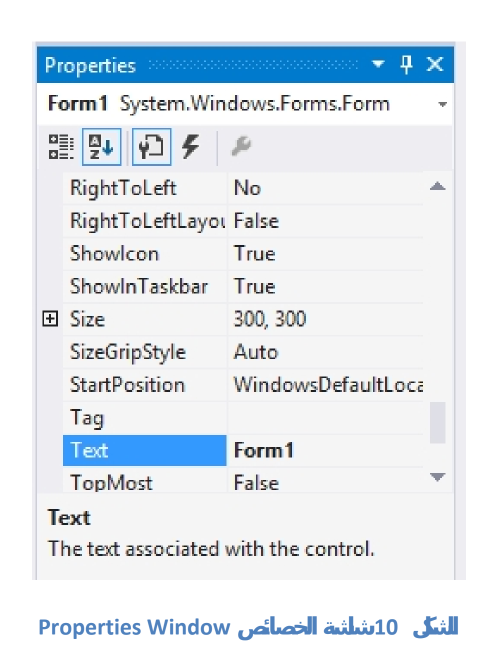
وهو الجزء الذي يعرض كافة مكونات المشروع من ملفات ومجلدات وواجهات وغيرها، كل شاشة جديدة أو عنصر جديد تضيفه إلى مشروعك سيظهر في هذا الجزء، ويمكنك كذلك فتح ملفات المشروع منه أو التحكم فيها كإعادة تسميتها أو حذفها أو تحديث مكونات المشروع أو عرض جميع الملفات...إذا لم يظهر لك متصفح المشروع فقم بالذهاب إلى القائمة View واخاره من هناك أو اضغط على الاختصار Ctrl+Alt+L وهذه صورة لمتصفح المشروع:



الشكل 9 متصفح المشروع Solution Explorer

3. شاشة الخصائص Properties Window:

تسمح لنا هذه الشاشة بالتحكم في خصائص الأدوات التي نستخدمها في مشروعنا، كأن نغير نوع الخط أو لونه، أو حجم الأدوات أو موقعها، ومن خلالها أيضا نستطيع الولوج إلى الأحداث Events المرتبطة بكل أداة، إذا لم تكن ظاهرة لديك في بيئة التصميم فقم بالذهاب إلى القائمة View واختر منها Properties Window أو اضغط على المفتاح F4 وهذه صورة لشاشة الخصائص:

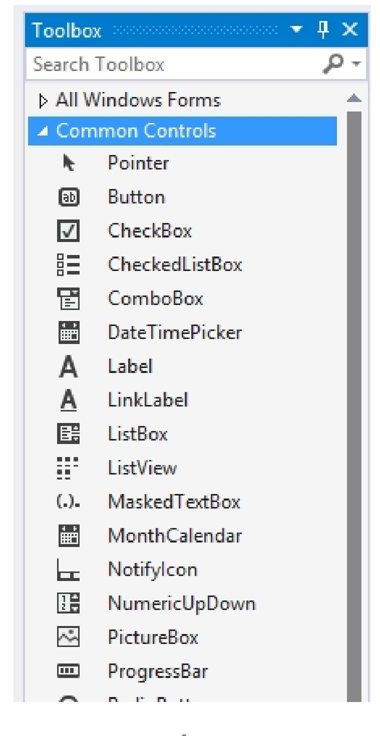


الشكل 10شاشة الخصائص Properties Window

4. علبة الأدوات ToolBox:

وتشمل كل الأدوات التي يمكننا استخدامها في برامجنا من أزرار ومربعات نصوص وقوائم منسدلة وغيرها من الأدوات، فقط نقوم بإختيار الأداة ومسكها وسحبها على الفورم من أجل استخدامها في برنامجنا، إن لم تكن ظاهرة عندك فاذهب إلى القائمة

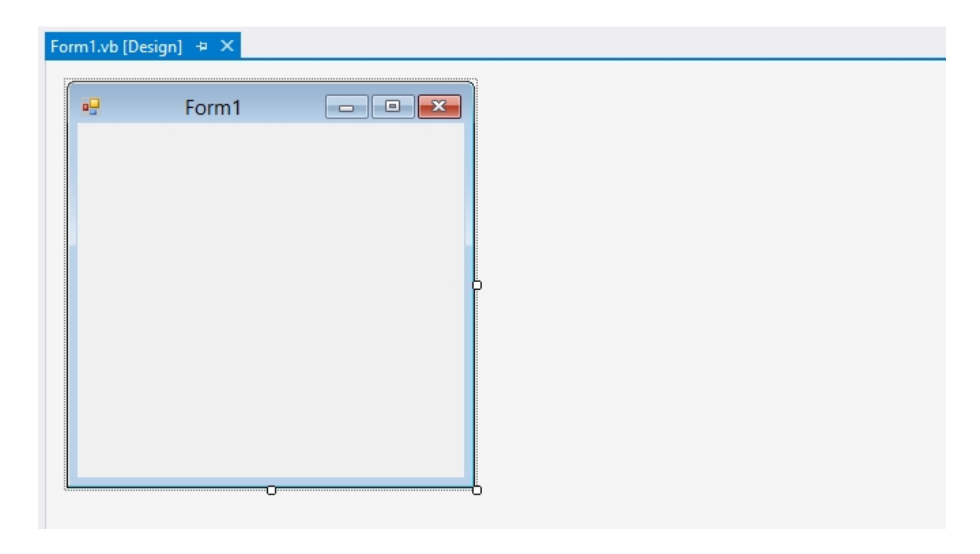
View ثم اخترها لكي تظهر، أو اكتف بالضغط على الإختصار Ctrl+Alt+X، وهذه صورة لعلبة الأدوات:



الشكل 11 علبة الأدوات ToolBox

5. شاشة التصميم:

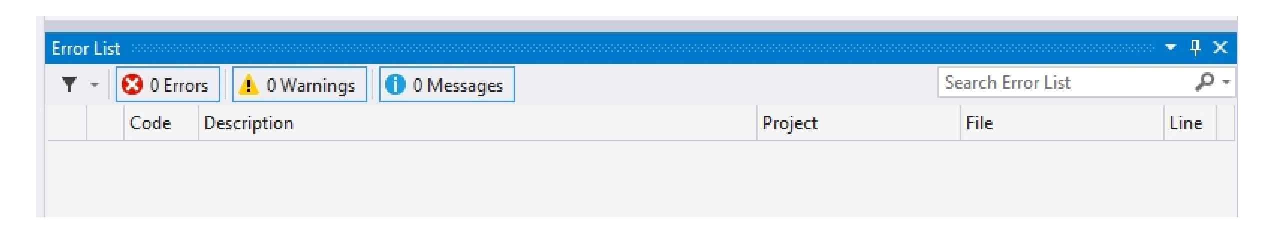
تعرض لنا هذه الشاشة الفورم الذي نريد تصميمه في البرنامج، والذي يمكننا تعديل خصائصه من شاشة الخصائص كلون الخلفية أو عنوان الفورم...، ويمكننا كذلك وضع أدوات عليه عبر سحبها من علبة الأدوات، ويمكننا أن نرى الفورم الذي يظهر في شاشة التصميم في مستعرض الملفات مع إمكانية حذفه أو إعادة تسميته...وهذه صورة لواجهة التصميم الافتراضية:



الشكل 12 واجهة التصميم Form Design

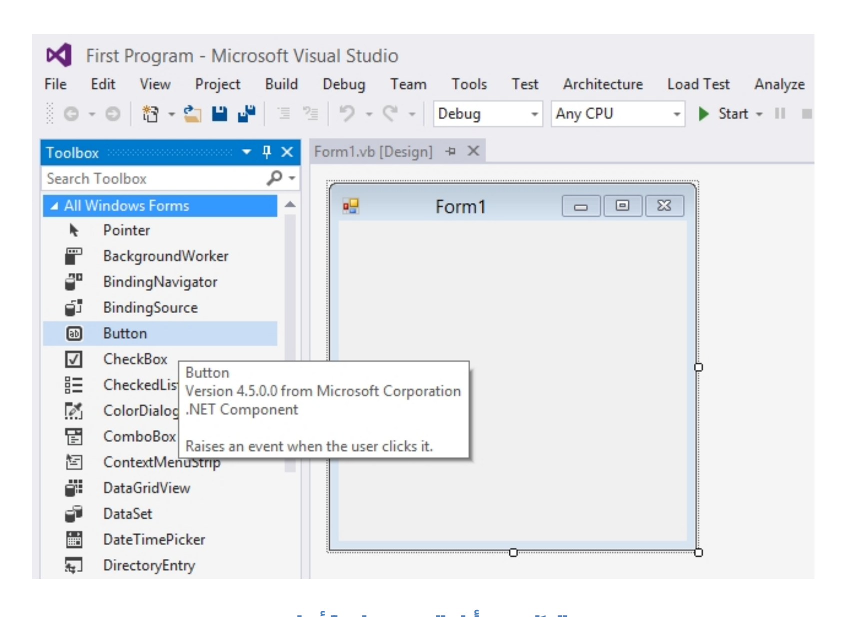
6. قائمة الأخطاء Error List:

وتعرض هذه النافذة الأخطاء المرتكبة قبل بدء عملية التنفيذ، من خلالها يمكنك معرفة مكان الخطأ ليتأتى لك تصحيحه. وهذه صورة لواجهة الأخطاء:

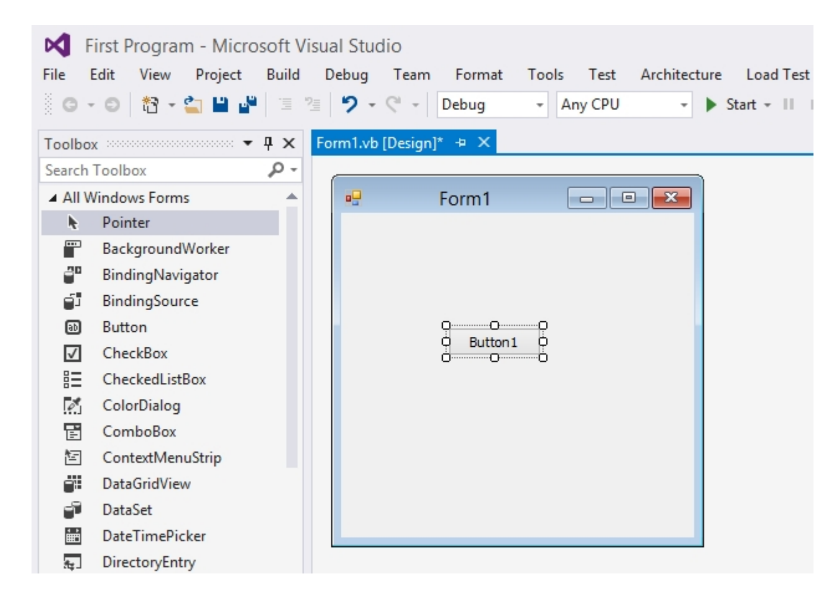


الشكل 13 قائمة الأخطاء Error List

سنقوم الآن بالذهاب إلى علبة الأدوات، واختيار أداة الزر Button الموجودة في التبويب Common Controls وكذلك في التبويب All Windows Forms، ونقوم بالضغط عليها وجرها إلى الفورم الظاهر في شاشة التصميم:

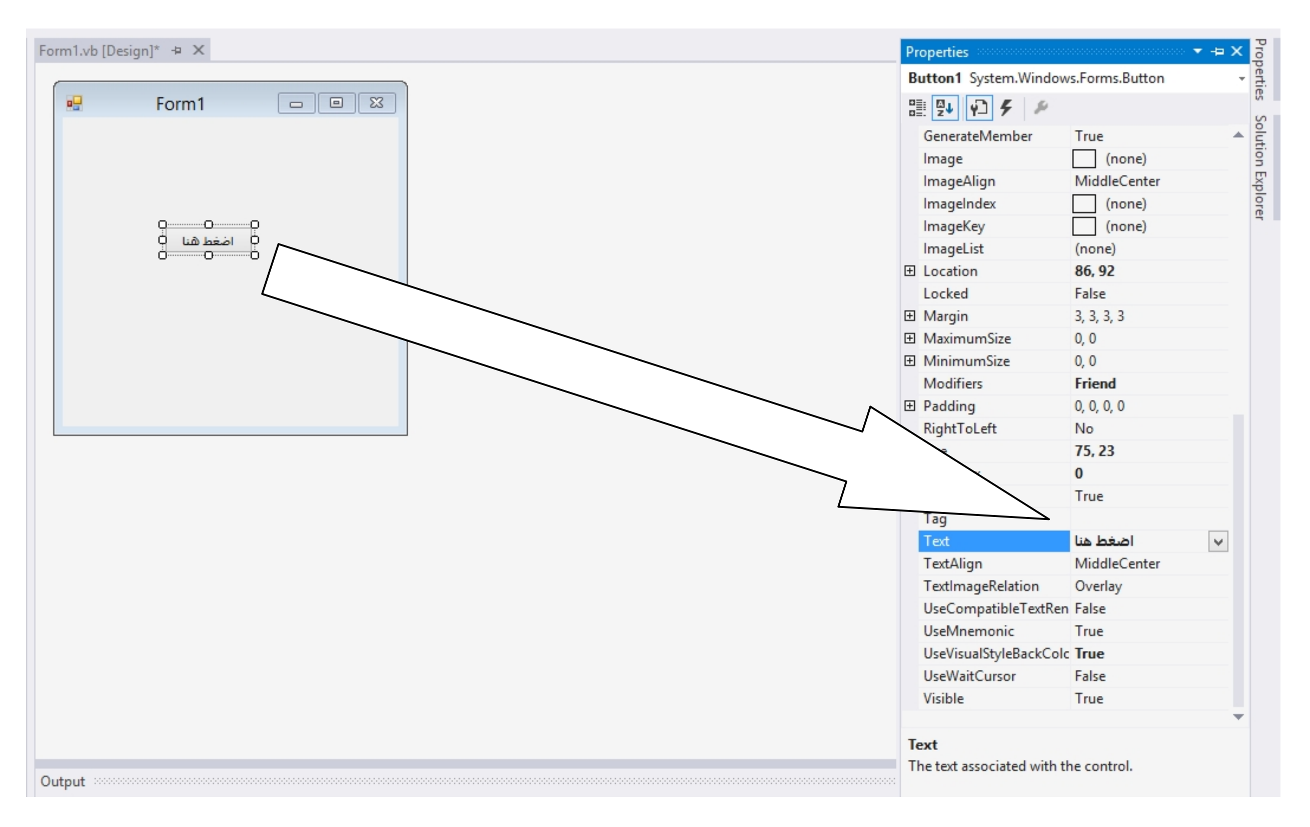


الشكل 14 أداة الزر في علبة الأدوات



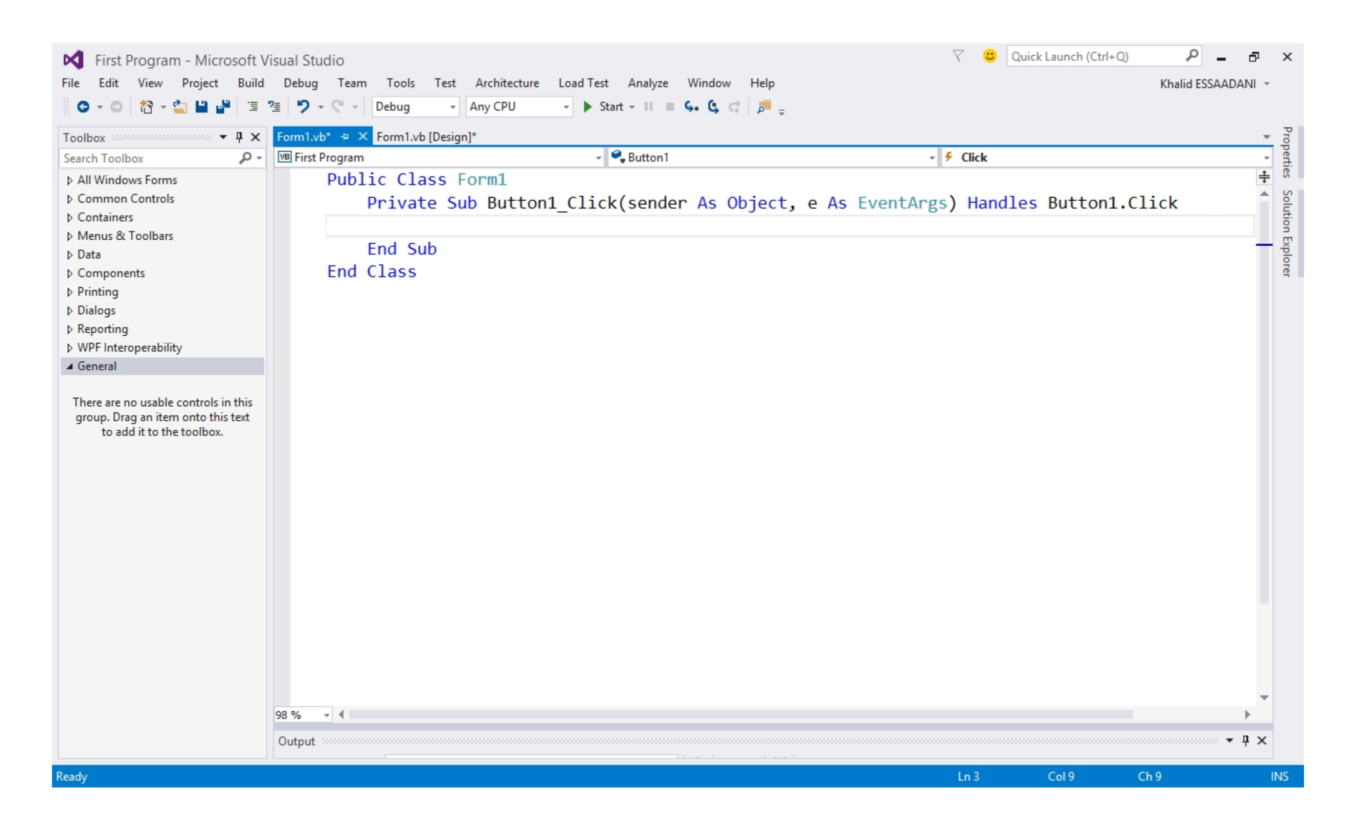
الشكل 15 أداة الزر بعد سحبها على الفورم

بعد وضع الزر على الفورم، نقوم بتغيير النص الظاهر عليه من شاشة الخصائص Properties Window، إذا لم تظهر لديك شاشة الخصائص بعد تحديد الزر، قم بالضغط على المفتاح F4 من لوحة المفاتيح، ثم ابحث عن الخاصية Text، واكتب فيها النص الذي تريد إظهاره على الزر وليكن مثلا "اضغط هنا".



الشكل 16 تغيير نص الزر من شاشة الخصائص

نريد من البرنامج أن يظهر لنا رسالة عند الضغط على هذا الزر، لعمل ذلك، نعمل دابل كليك على الزر ليتم نقلنا مباشرة إلى محرر الأكواد الذي تعرضه الصورة التالية:



السطر الأول يقوم بالإعلان عن فئة جديدة Class تحمل اسم Form1 وهو اسم الواجهة الافتراضية لدينا:

```
Public Class Form1
```

سنتعرف على مفهوم الفئات Classes في الفصول القادمة بالتفصيل إن شاء اللّه.

في السطرين المواليين تم إنشاء وظيفة Method سيتم تنفيذها عند الضغط على الزر، أي إصدار الحدث Click:

```
    Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click

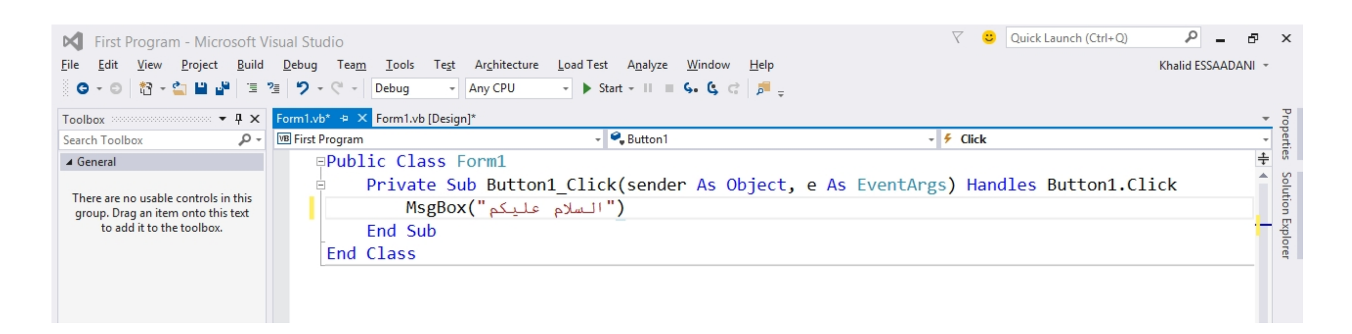
    End Sub
```

سنتعرف على مفهوم الوظائف Methods والأحداث Events بالتفصيل في الفصول القادمة بحول اللّه.

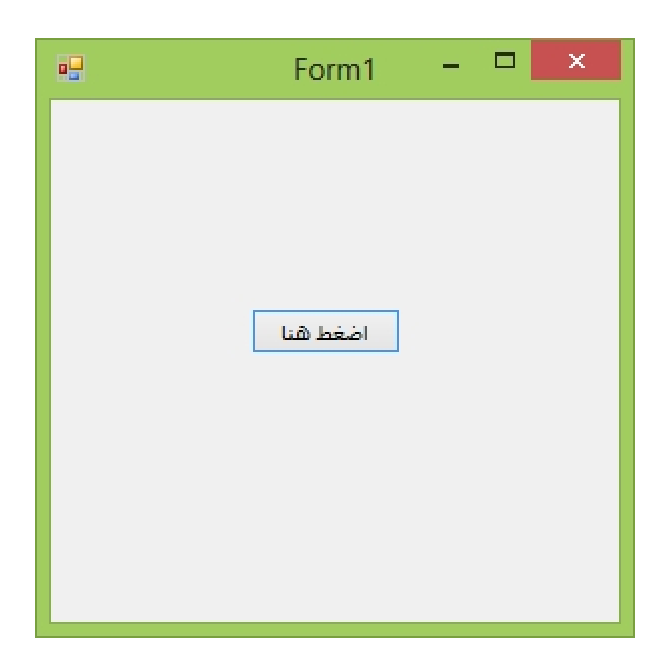
قبل الأمر End Sub نقوم بكتابة الشفرة التالية التي تعرض لنا رسالة" السلام عليكم":

```
MsgBox("السلام عليكم")
```

الدالة MsgBox تقوم بعرض رسالة للمستخدم بالنص الذي نعطيه لها.



الآن بقي علينا فقط أن ننفذ البرنامج ونشاهد كيف سيظهر، لعمل ذلك اضغط على المفتاح F5 أو اذهب بكل بساطة إلى الأمر Start بجانب المثلث الأخضر Start واضغط عليه، وانتظر قليلا ليظهر الفورم كما يلي:

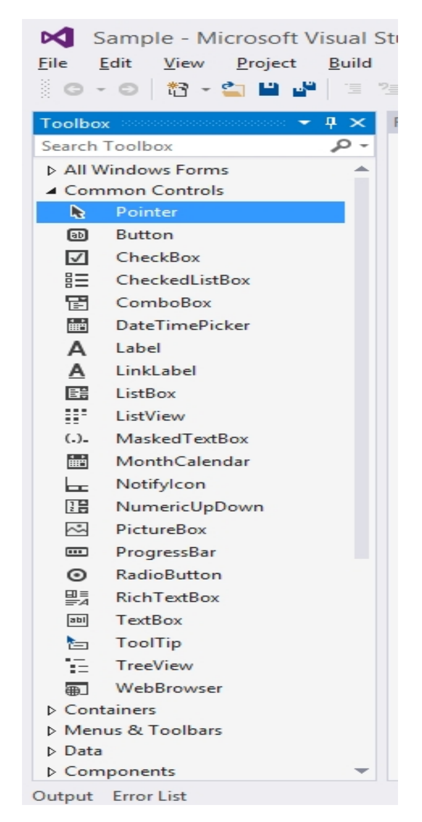


عند الضغط على الزر ستطالعك الرسالة التالية:



هكذا نكون قد أنجزنا أول برنامج لنا في لغة الفيجوال بيسك، في الفصول القادمة سوف نتعرف بالتفصيل على كيفية كتابة الأكواد ومكان كتابتها.

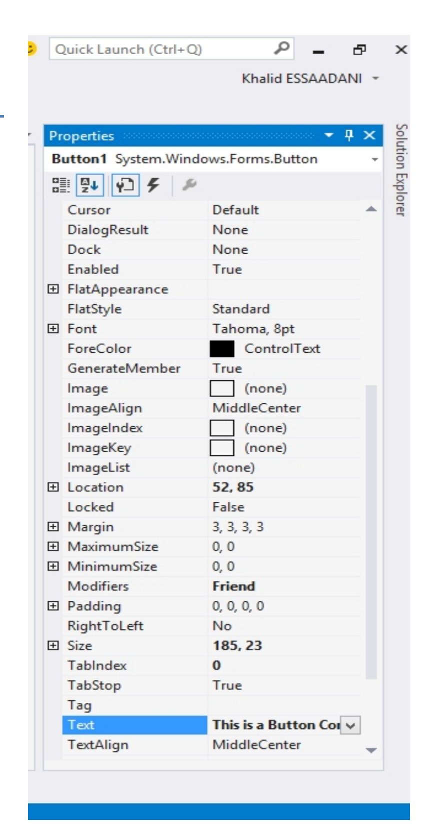
الأدوات Controls:



الأدوات هي كل ماستحتاجه لبناء برنامجك، من فورم Form وعلب النص Textbox وأزرار Button وقوائم ListBox وغيرها، وقد رأينا أين توجد هذه الأدوات وقلنا بأنها موجودة في علبة الأدوات Toolbox.

أفردنا كتابا خاصا بشرح الأدوات المتاحة في لغة الفيجوال بيسك، هذا الكتاب هو الجزء الثالث من الحقيبة البرمجية (المرحلة الأولى) ستجده في نفس مجلد هذا الكتاب.

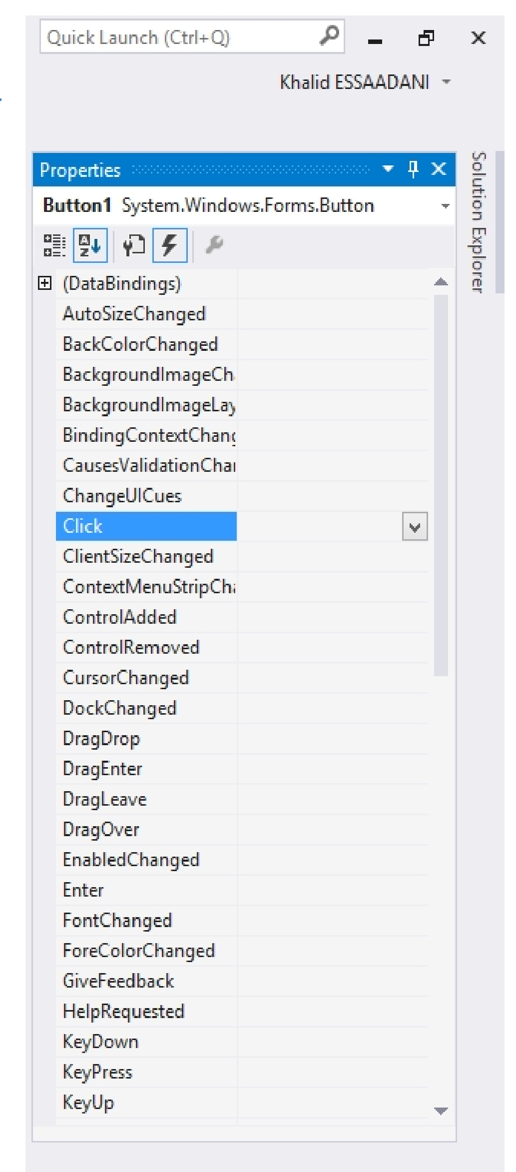
الخصائص Properties



الخصائص هي ما يميز شكل الأدوات، مثلا اللون، نوع الخط، الحجم، صورة الخلفية، الموقع على الشاشة، وغير ذلك من التفاصيل التي يمكنك مشاهدتها في واجهة الخصائص عند تحديد أداة معينة والضغط على الزر F4.

لنقرب مفهوم الخصائص إلى أذهاننا، سنأخذ على سبيل المثال بيتا، فالبيت هو بمثابة أداة Control وخصائصه

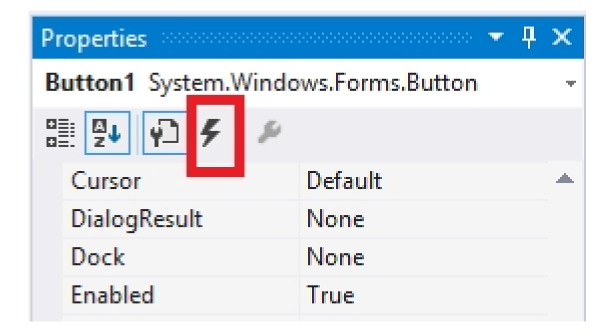
عديدة كلون البيت، ومساحته، وعلوه، و عدد أطباقه وما إلى ذلك، نفس الشيء ينطبق على أداوت البرمجة، فكل أداة تتوفر على خصائص تتميز بها عن غيرها، إضافة إلى أنها تشترك مع باقي الأدوات في مجموعة من الخصائص.



الأحداث Events

الأحداث هي أفعال تقع على الأدوات في لحظة معينة وتؤدي إلى تنفيذ إجراء مرتبط بها، مثلا حينما نقوم بالضغط على زر معين موجود على الفورم، فإن ذلك يتسبب في توليد حدث يسمى Click وهو مرتبط بإجراء Procedure سيتم تنفيذها في كل مرة يصدر هذا الحدث.

لكل أداة من أدوات الفيجوال بيسك العديد من الأحداث، يمكنك استعراضها من خلال تحديد أداة معينة، والذهاب إلى واجهة الخصائص ثم الضغط على الأيقونة التالية:



للدخول إلى حدث ما، يكفي أن تنقر عليه مرتين وسيتم نقلك إلى محرر الكود، وتحديدا إلى الإجراء المرتبط به، أي كود ستكتبه داخل هذا الإجراء سيتم تنفيذه أثناء إصدار الحدث المرتبط به:

التعليقات Comments

التعليقات هي عبارات نقوم بكتابتها في برامجنا ويتجاهلها المترجم Compiler لأن دورها يكون فقط هو عنونة وتوثيق Documentation الكود لتسهيل قراءته أو لتدوين بعض الملاحظات عليه من قبل المبرمج.

في لغة الفيجوال بيسك، يمكننا كتابة التعليقات في الكود عبر بدء السطر برمز المزدوجة الأحادية '، وهذه نماذج على بعض التعليقات المتنوعة:

```
'هذا تعليق مكتوب في سطر واحد

'أما هذا
'فهو تعليق متعدد
'الأسطر
```

مفهوم المتغيرات Variables

لنفترض أن العميل طلب منك برنامجا يقرأ رقمين، ثم يطبع أكبرهما (بمعنى أنه يريد منك أن تنشىء برنامجا يطلب من المستخدم أن يدخل رقمين كتابة من لوحة

المفاتيح أو من أي وسيلة إدخال أخرى، فيستقبل البرنامج هذين الرقمين ويقارنهما داخليا ويطبع على الشاشة Screen قيمة أكبر رقم).

كما تظهر الصورة التالية:

البحر الشاسع لدخول الخوارزميات من بابها الواسع

أدخل الرقم الأول:

أدخل الرقم الثاني:

ماهو الرقم الأكبر ؟

أول شيء ينبغي أن تفكر فيه، هو أنك ستحتاج إلى عددين للقيام بعملية المقارنة، وهذان العددان غير ثابتين و غير معروفين مسبقا، لأن المستخدم هو الذي سيدخلهما في البرنامج، وأنت لا يمكنك التكهن بهما، لذا فإنك ستحتاج إلى حاويات في الذاكرة تضع فيها القيم المدخلة لكي تقارنها في برنامجك، وللولوج إلى هذه الحاويات فأنت تحتاج إلى إعطائها أسماء مميزة وغير متشابهة، وكذلك تحتاج إلى تحديد نوع القيمة المراد تخزينها في كل حاوية.

القيم التي نجهلها ولا ندري قيمتها مسبقا، ونحتاج إلى تحديد نوع لها وتخزينها في الذاكرة تسمى متغيرات أو متحولات Variables، أي أن قيمتها ليست ثابتة، وأنها قابلة للتغيير المستمر أثناء تنفيذ البرنامج Runtime.

إذن، فالمتغيرات هي قيم يحتاجها البرنامج الذي ننشؤه من أجل إنجاز عمليات معينة عليها، والمستخدم هو من يقوم بإدخال هذه القيم إلى البرنامج (عبر لوحة المفاتيح مثلا) أو البرنامج نفسه يقوم ببعض العمليات ويحتاج إلى تخزين قيم مؤقتة في الذاكرة، ويمكن للقيم المدخلة أن تكون من أنواع شتى، إما رقمية، أو نصية، أو عبارة عن تاريخ،...

الإعلان عن المتغيرات

الإعلان عن متغير يعني حجز مكان في الذاكرة من أجل تخزين قيمة معينة قابلة للتغير في زمن التنفيذ Runtime (المقصود بزمن التنفيذ الوقت الذي يكون البرنامج في شغالا)، في لغة الفيجوال بيسك يمكننا الإعلان عن المتغيرات بالكلمة Dim كما يعرض المثال التالي:

```
Dim VARIABLE_NAME As DATA_TYPE
```

نبدأ أولا بالكلمة Dim التي تعني أننا بصدد الإعلان عن متغير ثم نقوم بكتابة اسمه مكان VARIABLE_NAME ثم نضع الكلمة AS وبعدها نوع البيانات المراد تخزينها في هذا المتغير، مثلا بيانات نصية، رقمية...

وهذه أمثلة لكيفية الإعلان عن مجموعة من المتغيرات:

```
Dim myNumber As Integer 'متغير رقمي من نوع صحيح طبيعي
Dim myDouble As Double 'متغير رقمي من نوع عشري
Dim myChar As Char 'متغير من نوع حرفي
Dim myString As String 'متغير من نوع نصي
Dim myDateTime As DateTime 'متغير من نوع تاريخ ووقت
```

```
Dim myBool As Boolean 'متغير من نوع منطقي
```

في المثال أعلاه قمنا بالإعلان عن عدة متغيرات، لكننا لم نعطها قيما بدئية، بمعنى أن أماكن هذه المتغيرات في الذاكرة ما تزال فارغة، ويلزمنا إعطاؤها قيما لكي نستخدمها، وهذه القيم إما أن تكون بدئية يحددها المبرمج في بداية الإعلان عن المتغيرات، أو تكون خلال مرحلة تنفيذ البرنامج بحيث تحصل المتغيرات على قيمها من خلال مدخلات المستخدمين.

أيضا يمكننا الاعلان عن مجموعة من المتغيرات ذات نفس نوع البيانات في سطر واحد، كما يعرض المثال الآتي:

```
Dim myNumber1, myNumber2, myNumber3 As Integer
```

أنواع البيانات Data Types

رأينا قبيل قليل مفهوم أنواع البيانات، وذكرنا أنها تمثل نوع القيم التي نريد تخزينها في المتغيرات التي نعلن عنها في برامجنا، بحيث لو كانت القيمة رقمية نستخدم نوع بيانات رقمي Integer وإن كانت نصية استخدمنا نوع بيانات نصي String وهكذا دواليك... هذه القيم يتم تخزينها في الذاكرة أثناء تنفيذ البرنامج بواسطة Common Language Runtime (CLR).

الآن بحول الله سوف نتبحر في أعماق أنواع البيانات لكي نتعرف عليها أكثر:

بالنسبة للقيم الرقمية، سوف نتعامل مع قسمين من أنواع البيانات الرقمية: وهي الأنواع الرقمية الصحيحة الطبيعية Integer والأنواع الرقمية العشرية Floating-Point.

الأنواع الصحيحة الطبيعية Integer تشمل الأرقام التي لا تحتوي على فاصلة عشرية، وتنقسم في لغة الفيجوال بيسك إلى عدة أقسام حسب المجال الرقمي الذي تختص به، فهنالك مثلا نوع البيانات الرقمي Byte الذي يشمل الأرقام الصحيحة الموجودة في المجال من 0 إلى 255، ويوجد النوع الرقمي Short الذي يبدأ من العدد 768 32- إلى غاية العدد 32 767، وهكذا دواليك. بينما تشمل الأنواع العشرية Floating-Point كل الأنواع الرقمية بما فيها تلك التي تحتوي على فاصلة عشرية ومن بين الأنواع العشرية المستخدمة في لغة الفيجوال بيسك النوع Double.

في المثال التالي سنقوم بالإعلان عن مجموعة من المتغيرات الرقمية الصحيحة الطبيعية والعشرية:

```
Dim IntegerNumber As Integer
Dim ByteNumber As Byte
Dim SByteNumber As SByte
Dim ShortNumber As Short
Dim LongNumber As Long
Dim SingleNumber As Single
Dim UIntegerNumber As UInteger
Dim ULongNumber As ULong
Dim UShortNumber As UShort
```

ملحوظة 1:

بعض الأنواع الرقمية أعلاه ستجدها مسبوقة بالحرف U مثل UInteger و ULong وهي اختصار ل Unsigned Integer و Unsigned Long إلخ..

الكلمة Unsigned تعني أن هذا النوع الرقمي خاص باحتواء القيم الإيجابية فقط، بمعنى لو أنك ستحتاج إلى متغير لتخزين القيم الإيجابية فقط دون القيم السلبية، فقم بالإعلان عنه من نوع Unsigned لأنه سيأخذ نفس مساحة التخزين في الذاكرة التي يأخذها النوع الأصلي إضافة إلى أنه يتيح لك مجالا رقميا يضاعف المجال الرقمي الخاص بالنوع الأصلي.

مثلا النوع Short يسمح باستعمال المجال الرقمي من من 32,768 - إلى 32,767، لو استعملنا النوع Unsignedمن هذا النوع سيسمح لنا بتخزين ضعف هذا المجال لأنه لا يقبل القيم السلبية وبالتالي سيتيح لنا مجالا رقميا أكبر للاعداد الإيجابية، أي أن النوع UShort سيسمح لنا بتخزين القيم الرقمية من 0 إلى 65,535.

ملحوظة 2:

ستجد في لغة الفيجوال بيسك بعض الأنواع الرقمية الأخرى مثل Int16 و Int32 و Int64، وفي الحقيقة هي ليست أنواع مختلفة بل هي تسميات أخرى لبعض الأنواع الرقمية التي أوردنا في المثال أعلاه، على سبيل المثال النوع Int32 هو نفسه النوع Integer وهذا الجدول يبين هذا الأمر:

النوع الرقمي	النوع المُوافق له
Int16	Short
Int32	Integer
Int64	Long

أما الأنواع النصية، فيوجد لدينا النوع String الذي يمكننا تخصيصه لتخزين القيم النصية، ويوجد كذلك النوع Char الذي يسمح بتخزين رمز واحد وليس سلسة نصية.

فيما يلي جدول يوضح مختلف أنواع البيانات المتوفرة في لغة الفيجوال بيسك:

النوع	مساحة التخزين	مجال القيم
Boolean	2 بايت	يقبل القيمتين True وFalse فقط.
Byte	1 بايت	من 0 إلى 255
SByte	1 بايت	من -128 إلى 127
Char	2 بايت	أي رمز أحادي Unicode Symbol
Date	8 بايت	من 1 يناير 0001 إلى 31 ديسمبر 9999
Decimal	16 بايت	
Double	8 بايت	
Integer	4 بايت	من -2,147,483,648 إلى 2,147,483,647
UInteger	4 بايت	من 0 إلى 4,294,967,295
Long	8 بايت	من -9,223,372,036,854,775,808 إلى 9,223,372,036,854,775,807
ULong	8 بايت	من 0 إلى 18,446,744,073,709,551,615
Object	4 بايت	القيم من جميع الأنواع ممكن تخزينها في النوع Object
Short	2 بايت	من 32,768 - إلى 32,767
UShort	2 بايت	من 0 إلى 65,535
Single	4 بايت	من -3.4028235E38 إلى 3.4028235E38
String	غالبا 2 بايت لكل رمز	من 0 إلى 2 بليون رمز أحادي Unicode Character

إسناد القيمة للمتغير

لإسناد قيمة بدئية لأي متغير، فإن الطريقة تكون عبر استخدام رمز يساوي (=) ثم بعده القيمة المراد إعطاؤها للمتغير، ويمكن عمل ذلك في زمن الإعلان، وكذلك بعد الإعلان، وهذه أمثلة توضح هذا الكلام:

```
Dim myNumber1 As Integer = 156
Dim myNumber2 As Integer
myNumber2 = 245
```

في السطر الأول أعطينا المتغير myNumber1 القيمة 156 في نفس وقت الإعلان، وفي السطر الثاني قمنا بإعطاء قيمة للمتغير myNumber2 بعد الإعلان عنه، لاحظ أن المتغيرات الرقمية تستقبل القيم بشكل مباشر، بينما المتغيرات النصية فيلزمنا استخدام علامات التنصيص (" ") ووضع النص داخلهما، كما توضح الأمثلة الآتية:

```
Dim Name As String = "خالد السعداني"
Dim Religion As String
Religion = "مسلم"
```

ملحوظة:

يمكننا إسناد قيمة متغير إلى متغير آخر بنفس الطريقة:

```
Dim Number1 As Integer = 540
Dim Number2 As Integer
Number2 = Number1
```

أو هكذا باختصار:

```
Dim Number1 As Integer = 540
Dim Number2 As Integer = Number1
```

في المثال أعلاه تم نقل قيمة المتغير Number1 إلى المتغير Number2 وبالتالي ستصبح قيمة هذا الأخير هي 540.

الروابط / المعاملات:

الروابط أو المعاملات هي رموز نستخدمها لإجراء بعض العمليات على المتغيرات أو على القيم بصفة عامة مثل العمليات الحسابية، أو عمليات مقارنة القيم (تحديد القيمة الكبرى والقيمة الصغرى) وغير ذلك.

الروابط الحسابية أو الرياضية Arithmetic operators:

وهي روابط نستخدمها في برامجنا من أجل القيام بالعمليات الرياضية مثل الجمع والطرح...، وهذا جدول رموز الروابط الرياضية وأمامها الدور المنوط بها:

الرابط	دوره
+	الجمع
-	الطرح
*	الجداء
/	القسمة
\	القسمة الصحيحة الطبيعية

Mod	باقي القسمة
^	القوة

الروابط الظاهرة أعلاه هي رموز نستخدمها حينما نود القيام بعملية حسابية على قيمتين أو أكثر، وهذه أمثلة على استخدام الروابط أعلاه:

```
Dim FirstNumber As Integer = 20

Dim SecondNumber As Integer = 4
'رابط الجمع
Dim Sum As Integer = FirstNumber + SecondNumber

'رابط الطرح
Dim Dif As Integer = FirstNumber - SecondNumber

'رابط الضرب
Dim Mul As Integer = FirstNumber * SecondNumber

'رابط القسمة
Dim Div As Integer = FirstNumber / SecondNumber

'رابط القسمة الصحيحة
Dim IntDiv As Integer = FirstNumber \ SecondNumber

'رابط باقي القسمة
Dim Modulus As Integer = FirstNumber Mod SecondNumber

'رابط القوة
Dim Exp As Integer = FirstNumber ^ SecondNumber
```

روابط دمج النصوص String Concatenation operators:

ويمكننا هذا النوع من الروابط من دمج نصين أو أكثر مع بعضهم البعض، ويكون من خلال استخدام الرمز + أو &، وهذا مثال على دمج قيمتين نصيتين:

```
Dim FirstString As String = "Think First "
Dim SecondString As String = "Code Later"
Dim ResultString As String = FirstString & SecondString
'أو
Dim ResultString As String = FirstString + SecondString
```

المتغير ResultString سيحتوي على نتيجة دمج قيمتي المتغيرين المتقدمين، أي أن محتواه سيكون هو: Think First Code Later.

روابط المقارنة Comparison operators:

وهي روابط نستخدمها من أجل مقارنة قيمتين وتحديد نوع العلاقة بينهما (أكبر من، أصغر من، تساوي، ...) ونتيجة المقارنة تكون منطقية boolean، أي لا تقبل إلا قيمتين: إما صحيح true وإما خطأ false، فيما يلي جدول يعرض رموز الروابط المستخدمة في عمليات المقارنة:

الرابط	دوره
>	أكبر من
<	أصغر من
=	يساوي
<>	يخالف
>=	أكبر من أو يساوي
<=	أصغر من أو يساوي

وهذه أمثلة على استخدام هذا النوع من الروابط:

```
'النتيجة خاطئة لأن 4 ليست أكبر من 5
Dim Value1 As Boolean = 4 > 5

'النتيجة صحيحة لأن 4 أصغر من 5
Dim Value2 As Boolean = 4 < 5
```

```
'5 لأن 4 لا تساوي 5 النتيجة خاطئة
Dim Value3 As Boolean = 4 = 5

'5 لأن 4 تخالف 5 النتيجة صحيحة
Dim Value4 As Boolean = 4 <> 5

'5 النتيجة صحيحة لأن أصغر من أو يساوي
Dim Value5 As Boolean = 4 <= 5

'5 النتيجة خاطئة لأن 4 ليست أكبر من أو يساوي
Dim Value6 As Boolean = 4 >= 5
```

روابط إسناد القيمة Assignment Operators:

وهي رموز نستخدمها من أجل تخزين وإسناد قيمة إلى متغير ما، وقد رأينا في إحدى الفقرات السابقة أن أبرز رابط يستخدم لإسناد القيم هو الرابط يساوي (=) وهنا قائمة لبعض الرموز التي تستخدم معه من أجل تخزين القيم وحسابها في نفس الوقت:

الرابط	دوره
=	الجمع
^=	إسناد القيمة بعد حساب القوة
*=	إسناد القيمة بعد حساب الجداء
/=	إسناد القيمة بعد حساب القسمة
\=	إسناد القيمة بعد حساب القسمة الصحيحة الطبيعية
+=	إسناد القيمة بعد حساب الجمع
-=	إسناد القيمة بعد حساب الطرح
&=	إسناد القيمة بعد دمجها بقيمة معينة

قد يلوح لك شيء من الغموض في الروابط أعلاه وهذا طبيعي جدا لذا سنعرض في المثال الآتي نموذجا لكيفية استخدام كل هذه الروابط التي نستخدمها بغرض إسناد القيم في متغيرات معينة:

```
'تخزين القيمة 5 في المتغير
    myVariable = 5

سيتم حساب قيمة المتغير بقوة 2 أي 5 أس 2 وبالتالي ستصبح قيمته 25
    myVariable ^= 2

'سيتم ضرب قيمة المتغير في 2 وتصبح قيمته 25 في 2 أي 50
    myVariable *= 2

'سيتم قسمة قيمة المتغير على 2 وتصبح قيمته 50 على 2 أي 25
    myVariable /= 2

'سيتم قسمة قيمة المتغير قسمة صحيحة على 2 أي ستصبح قيمته 25 على 2 أي 12
    myVariable \= 2

'سيتم إضافة 2 إلى قيمة المتغير وتصبح قيمته 12+2 أي 14
    myVariable += 2

'سيتم طرح 2 من قيمة المتغير وبالتالي تصبح 14 – 2 أي 12
    myVariable -= 2

'سيتم دمج قيمة المتغير مع القيمة 7 وتصبح قيمته 12 مدموجة مع 7 أس 127
    myVariable &= 7
```

الروابط المنطقية Logical operators:

هي روابط نستخدمها من أجل الحصول على نتيجة شرطين أو أكثر، والنتيجة تكون منطقية Boolean، أي لا تقبل إلا قيمتين: إما صحيح True وإما خطأ False، هذه الروابط هي AND والذي يعني (و) وتكون النتيجة صحيحة إذا كانت كل أطراف

الشرط صحيحة، والرابط OR والذي يعني (أو) وتكون النتيجة صحيحة إذا كان هنالك طرف واحد فقط أو أكثر في الشرط صحيحا.

حتى نبسط مفهوم الروابط المنطقية، دعنا نستحضر مثالا سهلا، لو قلت لك: سيزورني أحمد و كمال، فذلك يعني أن **كلامي سيكون صحيحا حينما سيأتيان معا**، لكن لو قلت لك: قد يزورني أحمد أو كمال، فالمعنى مختلف **وكلامي سيكون صحيحا سواء حضر أحمد أو كمال**.

الآن لاحظ معي هذا المثال لتتعرف على كيفية استخدام الروابط المنطقية:

```
' العبارتان صحيحتان معا إذن المتغير قيمته صحيحة
    Dim Value1 As Boolean = (3 < 4 And 7 > 5) 'Value1 = True

'إحدى العبارات خاطئة (5 تساوي 8) إذن المتغير قيمته خاطئة
    Dim Value2 As Boolean = (10 > 9 And 5 = 8) 'Value2 =
False

'إحدى العبارتين صحيحة وبما أننا نستخدم الرابط أو فالنتيجة صحيحة
    Dim Value3 As Boolean = (8 < 4 Or 6 > 2) 'Value3 = True

'العبارتان معا خاطئتان إذن النتيجة خاطئة لأننا نستخدم الرابط أو
    Dim Value4 As Boolean = (1 > 9 Or 6 = 3) 'Value4 = False

    ' العبارة الثانية صحيحة إذن النتيجة صحيحة لأننا
أو نستخدم الرابط
    Dim Value5 As Boolean = (1 > 9 Or 6 / 2 = 3) 'Value5 =
True
```

الروابط المنطقية المختصرة Short Circuited Operators:

هي روابط منطقية نستخدمها بغرض اختصار عملية التحقق بناء على شروط كل بنية من بنيات الروابط المنطقية، على سبيل المثال الرابط AND يتحقق من جميع العبارات فإن وجد عبارة واحدة لا تحقق الشرط يعيد لي القيمة خطأ False، فلو افترضنا أن لدينا العديد من العبارات الواجب التحقق من صحتها وأول عبارة تكسر القاعدة فالمفروض ألا يتم إكمال عملية التحقق لأنه خلاص بعدم تحقق عبارة واحدة فالنتيجة المعادة هي False حتى ولو كانت كل العبارات التي تأتي بعدها تعيد القيمة True، عند استخدام الرابط المنطقي AND فإن عملية التحقق تستمر مع كل العبارات حتى وإن وجدت عبارة تكسر القاعدة لهذا يحسن الاستعاضة عنها بالرابط الأحادي AndAlso الذي يوقف عملية التحقق بمجرد العثور على أول عبارة لا تحقق الشرط دون تضييع الوقت في التحقق من باقي العبارات على خلاف الرابط AND الذي يعمل ببلاهة ويمر على كل العبارات حتى ولو وجد عبارة سابقة تكسر القاعدة.

نفس الكلام ينطبق على الرابط المنطقي OR، فالقاعدة تقول أنه بوجود عبارة واحدة تعيد القيمة True تكون القيمة العامة المعادة هي True حتى وإن كانت كل العبارات الأخرى تعيد القيمة False، لكن لو افترضنا أن لدينا مئة عبارة يجب التحقق منها وكلها مرتبطة فيما بينها بالرابط OR، وأول عبارة من هذه العبارات تعيد القيمة True، المفروض أن تتوقف عملية التحقق من باقي العبارات وإرجاع القيمة True بعد أول تحقق، إلا أن هذا لا يقع للأسف وسيتم إكمال عمليات التحقق من جميع العبارات حتى وإن وجدت عبارة سابقة تعيد القيمة True، ولاجتياز هذه المشكلة يستحسن

استخدام الرابط المنطقي الأحادي OrElse الذي يوقف عمليات التحقق بعد العثور على أول عبارة تعيد القيمة True دون الحاجة إلى الانتقال إلى باقي العبارات للتحقق منها لأنه خلاص وجد عبارة تفي بالغرض.

فيما يلي أمثلة لاستخدام الروابط المنطقية الأحادية AndAlso و OrElse:

```
        'Value1 = False
        Dim Value1 As Boolean = (3 > 4 AndAlso 7 > 5 AndAlso 8
> 4)

        'Value2 = True
        Dim Value2 As Boolean = (5 > 9 OrElse 4 = 2 * 2 OrElse
6 < 4)
```

المتغير الأول Value1 ستكون قيمته False لأن أول عبارة غير صحيحة وبالتالي النتيجة غير صحيحة False، الرابط AndAlso سيوقف عمليات المقارنة الأخرى لأن القاعدة تحققت مع أول عبارة.

المتغير الثاني Value2 ستكون قيمته True لأن إحدى العبارات وهي العبارة الثانية 4 = 2 * 2 صحيحة وبالتالي نتيجة الشرط هي True، الرابط OrElse سيتوقف عند العبارة الثانية ويعيد القيمة True لأن القاعدة تحققت مع ثاني عبارة ولا حاجة للتحقق من العبارة الثالثة.

البنية الشرطية باستخدام الأمر If:

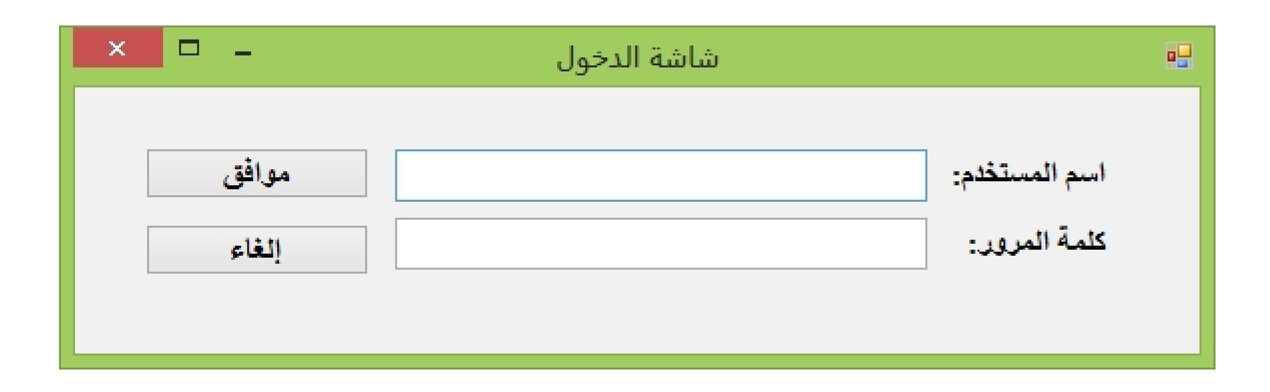
أحيانا نحتاج إلى إجراء تحقق من عبارة معينة في برنامجنا، فإن كانت نتيجة التحقق تساوي قيمة معينة ننفذ شفرة معينة، وإن كان العكس أو كانت تساوي قيمة أخرى ننفذ شفرات أخرى، على سبيل المثال، نريد إنشاء برنامج يتكون من شاشة لتسجيل الدخول مثل تلك التي في برنامج Skype أو برنامج Messenger، فنحن ملزمين بالتحقق من اسم المستخدم وكلمة السر المدخلين، إن كانا صحيحين سمحنا بعملية الدخول، وإن كان أحدهما خاطئا أظهرنا رسالة خطأ.

في البرمجة تسمى عملية التحقق من عبارة معينة شرطا Condition أو بنية شرطية Flow Control، وصيغتها كما يلي:

```
If 'الشرط الأول' Then
    'إن تحقق الشرط الأول نفذ الأوامر التالية'
ElseIf 'الشرط الثاني' Then
    'إن تحقق الشرط الثاني نفذ الأوامر التالية'
Else
    'إن لم يتحقق أي شرط مما سبق فنفذ ما يلي'
End If
```

الشرط الأول نضعه بعد الأمر If فإن تحقق يتم تنفيذ الكود الذي يليه، وإن لم يتحقق يتم التحقق من الشرط الذي يتبعه ويأتي بعد الكلمة ElseIf فإن تحقق يتم تنفيذ الكود الموالي له، وهكذا دواليك، فإن لم يتحقق أي شرط من الشروط يتم تنفيذ الأوامر الموجودة في البلوك Else.

الآن لنعطي مثالا نتعلم من خلاله طريقة استخدام البنية الشرطية في برامجنا لنفترض أننا بصدد إنشاء شاشة لتسجيل الدخول، مثل الشاشة الظاهرة في الصورة التالية:



في شاشة الدخول أعلاه، سيقوم المستخدم بكتابة اسمه، وبكتابة كلمة المرور، ونحن سنستقبل هذين القيمتين، وسنقوم بتخزينهما في متغيرين نصيين، ثم نقارنهما مع البيانات المسجلة عندنا في البرنامج أو المخزنة في ملف أو قاعدة بيانات، فإن كان هناك توافق بين البيانات المدخلة وبين البيانات المخزنة، نسمح بعملية الدخول، وإلا نظهر رسالة خطأ تعلم المستخدم بأن عملية تسجيل الدخول فشلت.

نأتي أولا ونقوم بتصميم شاشة الدخول، قم بإنشاء مشروع جديد من نوع Windows Forms Application، ثم اسحب داخل الفورم الأدوات التالية:

الأداة	اسمها	دورها
TextBox	txtUserName	من أجل ادخال اسم المستخدم
TextBox	txtPassWord	من أجل ادخال كلمة المرور
Label	lblUserName	من أجل عرض النص الخاص باسم المستخدم
Label	lblPassWord	من أجل عرض النص الخاص بكلمة المرور
Button	btnOK	من أجل التحقق من البيانات المدخلة

BuTTon	btnCancel	من أجل إلغاء العملية والخروج من الشاشة

بعد أن تقوم بتصميم الأدوات وتغيير أسمائها، انقر الآن على الزر btnOK وادخل إلى مكان الكود واكتب الكود التالي الذي يتحقق من البيانات المدخلة ويقارنها ببيانات الدخول المسجلة، فإن حصل توافق بينهما سمح بعملية الدخول وإلا فإنه يمنع العملية:

```
    Private Sub btnOK_Click(sender As Object, e As EventArgs)
Handles btnOK.Click

        Dim UserName As String = "myUserName"
        Dim PassWord As String = "myPWD123"

        If txtUserName.Text = UserName And txtPassWord.Text =
PassWord Then
            MsgBox("تمت عملية الدخول بنجاح")
        ElseIf txtUserName.Text <> UserName And
txtPassWord.Text = PassWord Then
            MsgBox("اسم المستخدم المدخل غير صحيح")
        ElseIf txtUserName.Text = UserName And txtPassWord.Text
<> PassWord Then
            MsgBox("كلمة المرور المدخلة غير صحيحة")
        Else
            MsgBox("رجاء البيانات المدخلة غير صحيحة")
        End If
    End Sub
```

في الكود أعلاه قمنا بالإعلان عن متغيرين UserName و PassWord، الأول قمنا بتخزين القيمة myUserName فيه، والثاني قمنا بتخزين القيمة myPWD123 فيه، ثم

بعد ذلك قمنا بالتحقق من القيم المدخلة في مربعات النصوص بالقيم التي قمنا بتخزينها في المتغيرين.

إن كان اسم المستخدم يساوي اسم المستخدم المسخل وكلمة المرور تساوي كلمة المرور المدخلة عرضنا رسالة: تمت عملية الدخول بنجاح.

إن كان اسم المستخدم مخالفا لاسم المستخدم المدخل وكلمة المرور تساوي كلمة المرور المدخلة عرضنا رسالة: اسم المستخدم المدخل غير صحيح.

إن كان اسم المستخدم يساوي اسم المستخدم المدخل وكلمة المرور تخالف كلمة المرور المدخلة عرضنا رسالة: كلمة المرور المدخلة غير صحيحة.

أما إن لم يتحقق أي شرط من الشروط السابقة فسوف يتم عرض الرسالة التي يأتت بعد الأمر Else والتي تقول: رجاء البيانات المدخلة غير صحيحة.

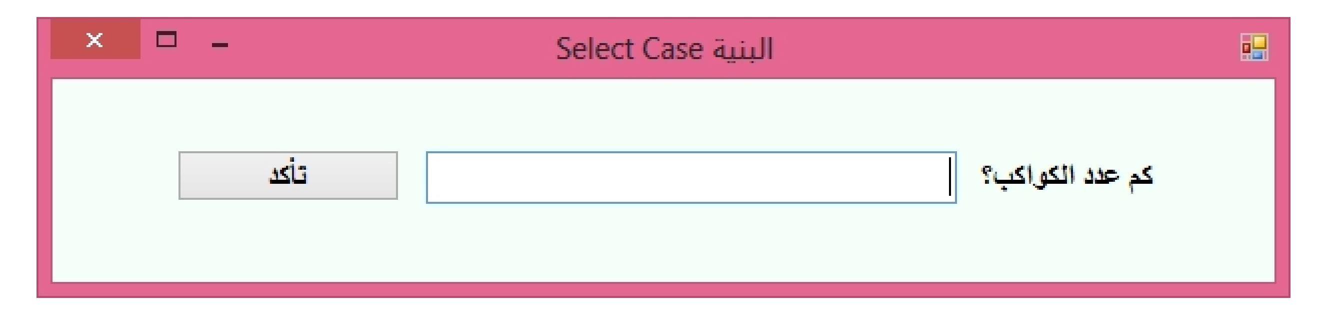
البنية الشرطية باستخدام الأمر Select Case:

الأمر Select Case يؤدي تقريبا نفس الدور الذي تؤديه البنية الشرطية باستخدام الكلمة If، وغالبا ما نستخدمه حينما يكون لدينا شروط قليلة ومحددة فنستعيض به عن الكلمة If.

```
Select Case 'العبارة المراد التحقق منها
    Case 'الحالة الأولى'
       'نفذ ما يلي'
    Case 'الحالة الثانية
       'نفذ ما يلي'
    Case 'الحالة الثالثة
       'نفذ ما يلي'
    Case Else 'إن لم تتحقق أي حالة من الحالات
       'نفذ ما يلي'
End Select
```

نضع بعد الأمر Select Case العبارة التي نريد التحقق من قيمتها، فإن كانت مساوية للحالة الأولى Case ننفذ الأوامر التي تليها، وإن كانت مخالفة للحالة الأولى وتساوي الحالة الثانية ننفذ الأوامر التي تتبع الحالة الثانية...إن لم تتحقق أية حالة من الهالات يتم تنفيذ الأوامر التي تأتي بعد الأمر Case Else.

سنقوم الآن بإنشاء برنامج يطلب من المستخدم إدخال عدد كواكب المجموعة الشمسية، وبناء على القيمة المدخلة نظهر له رسالة.



قم أولا بتصميم الفورم كما يظهر أعلاه وذلك بسحب الأدوات التالية:

الأداة	اسمها	دورها
TextBox	txtNumber	من أجل ادخال عدد الكواكب
Label	lblNumber	من أجل عرض السؤال
Button	btnOK	من أجل التأكد من الجواب المدخل

بعد ذلك انقر على الزر btnOK من أجل الانتقال إلى الحدث Click الخاص به، ثم اكتب الكود الآتي الذي يتحقق من الجواب المدخل ويقارنه بالعدد الصحيح لكواكب المجموعة الشمسية:

```
    Private Sub btnOK_Click(sender As Object, e As EventArgs)
Handles btnOK.Click

        'متغير رقمي يحتوي على العدد الصحيح
        Dim PlanetCount As Integer = 9

        'متغير يحتوي على القيمة المدخلة من طرف المستخدم
        Dim PlanetCountEnterred As Integer =
Val(txtNumber.Text)

        Select Case PlanetCountEnterred
            Case 9
                MsgBox("الجواب صحيح")
            Case Is < 8
                MsgBox("العدد المدخل أقل من العدد الصحيح")
            Case 10 To 70
                MsgBox("العدد المدخل يتراوح بين مجال غير صحيح")
            Case 8
```

```
                MsgBox("لقد اقتربت من الجواب الصحيح")
            Case Else
                MsgBox("القيمة المدخلة خاطئة")
        End Select
    End Sub
```

البنية الشرطية باستخدام الكلمة IIF:

الأمر IIF نستخدمه إذا أردنا تنفيذ أمر من بين أمرين في حال تحقق الشرط، وهو شبيه جدا بالرابط الثلاثي Ternary Operator في لغة سي شارب، وصيغته كما يلي:

```
IIf(الشرط, الأمر الأول, الأمر الثاني)
```

إذا تحقق الشرط سيتم تنفيذ الأمر الأول، وإذا لم يتم تحققه سيتم تنفيذ الأمر الثاني، وهذا مثال يعرض كيفية استخدام الأمر IIF:

```
        Dim Password As String = "myPWD123"
        Dim Access_State As String = IIf(Password = "myPWD123",
"كلمة المرور صحيحة", "كلمة المرور غير صحيحة")
```

المتغير Access_State سيحتوي على القيمة "كلمة المرور صحيحة" لأن قيمة المتغير Password تساوي "myPWD123" وبالتالي سيتم تنفيذ الأمر الأول، إن كانت قيمة المتغير Password مخالفة ل "myPWD123" سيتم تنفيذ الأمر الثاني وبالتالي تصبح قيمة المتغير Access_State هي "كلمة المرور غير صحيحة".

البنية التكرارية Loops

أحيانا نحتاج في تطبيقاتنا إلى تكرار أمر برمجي معين أكثر من مرة، وهذا لا يعني أن نعيد تنفيذ نفس الأمر فقط، بل يمكننا كذلك إجراء بعض العمليات على كل عنصر يصله التكرار، فلو افترضنا أنه لدينا قاعدة بيانات تحتوي على بيانات العملاء، وأردنا إضافة أرقام الهواتف لهم جميعا، ستلاحظ أن عملية إسناد هذا الحقل لكل عميل ستستنزف منك وقتا طويلا لا سيما وإن كان عدد العملاء كبيرا جدا، لذلك تعتبر الآليات التكرارية أو الحلقات هي أفضل خيار لتكرار تعليمات معينة بقدر محدد مع إمكانية إضفاء لمسة خاصة على كل عنصر تبلغه البنية التكرارية. أو لنفترض مثلا أننا بصدد برمجة تطبيق صغير يخزن قيمة رقمية في متغير معين ويطلب من المستخدم تخمين هذه القيمة، حتما نحن لا نعرف عدد الاحتمالات التي سيقوم بها المستخدم لكي يصل إلى نفس القيمة المخزنة عندنا، لذلك يتوجب طرح نفس السؤال في كل مرة يدخل المستخدم فيها قيمة مخالفة للقيمة المخزنة.

أو لنفترض أننا بصدد إنتاج برنامج بسيط يسأل المستخدم عن اسم أول خلفاء الدولة العباسية، فإن كان الجواب صائبا يتوقف البرنامج وإن كان الجواب خاطئا يعيد طرح السؤال على المستخدم إلى حين يحصل على الجواب الصحيح، في حالتنا هذه نحن لا نعلم عدد المرات اللازمة لكي يجيب المستخدم بالجواب الصائب، لذلك فنحن بحاجة إلى إجراء تكرار للأمر الذي يطبع السؤال في كل مرة يخطىء المستخدم في الإجابة.

توفر لنا لغة الفيجوال بيسك عدة أنواع من البنيات التكرارية، سنستعرضها بالترتيب في الفقرات القادمة.

الصيغة التكرارية الشرطية : For Next:

تمكننا البنية التكرارية For..Next من تكرار أمر برمجي معين لعدد مرات محدد، حيث تسمح لنا بتحديد نقطتي بداية ونهاية التكرار مع إمكانية تحديد معامل التدرج، وصيغتها كما يلي:

```
For Count = Start To End [Step Step]
        '[statements]
         [Exit For]
Next Count
```

حيث المتغير Count هو العداد الذي سيحدد عدد مرات تكرار الأمر برمجي، ويبدأ من القيمة الرقمية Start وينتهي بالقيمة الرقمية End مع إمكانية تحديد معامل التدرج (في كل تكرار بكم تزداد أو تنقص نسبة العداد افتراضيا تكون الزيادة بواحد) في الأمر Step، بعد ذلك نقوم بكتابة الأمر البرمجي الذي نود تكراره، ويمكننا الخروج من التكرار من خلال الأمر Exit For، في ختام البنية التكرارية نضع الكلمة Next متبوعة باسم العداد (اختيارية ويمكننا عدم كتابة اسم العداد)، وفيما يلي مثال بسيط يوضح كيفية استخدام هذا النوع من الحلقات:

قم بإنشاء مشروع من نوع Windows Forms Application، ثم اسحب على الفورم الأدوات التالية وغير خصائصها كما يلي:

الأداة	اسمها	دورها
ListBox	lbNumbers	من أجل عرض القيم الناتجة عن التكرار
Button	btnRepeat	من أجل تنفيذ أمر التكرار باستخدام الحلقات For..Next

قم بتصميم الفورم كما تعرض الصورة الآتية:



انقر على الزر btnRepeat مرتين من أجل الانتقال إلى الحدث Click الخاص به، واكتب الأمر التالي:

```
    Private Sub btnRepeat_Click(sender As Object, e As EventArgs) Handles btnRepeat.Click
        'متغير رقمي بمثابة عداد للحلقات
        Dim nCount As Integer

        For nCount = 0 To 10 Step 1
            Me.lbNumbers.Items.Add("العنصر:  " & nCount)
        Next

    End Sub
```

قمنا بالإعلان عن متغير رقمي اسمه Count ثم استعملناه في عملية التكرار ليبدأ من العدد 0 وينتهي بالعدد 10 مع الزيادة بواحد عند كل تكرار، وفي كل مرة يقوم التكرار بإضافة قيمة العداد إلى أداة LbNumbers مدموجة مع النص "العنصر:"، عند تنفيذ هذا الكود ينبغي أن نحصل على نتيجة مماثلة للصورة الآتية:



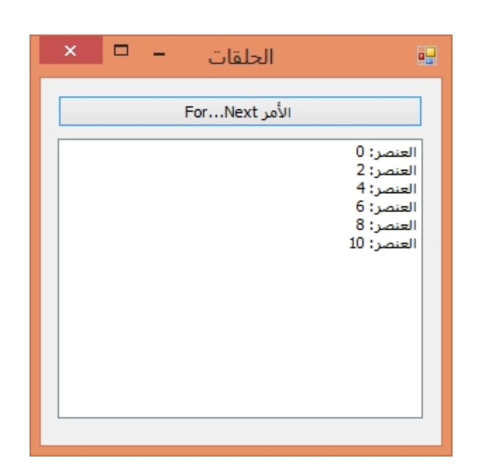
ملحوظة:

الأمر Step في المثال أعلاه لا دور له لأن نسبة الزيادة الافتراضية هي 1، ويمكننا استخدامه في حال أردنا تغيير قيمة التدرج، كما يوضح الكود الآتي الذي غيرنا فيه قيمة Step:

```
Dim nCount As Integer

For nCount = 0 To 10 Step 2
    Me.lbNumbers.Items.Add("العنصر:  " & nCount)
Next
```

هذه المرة جعلنا نسبة التدرج هي 2 وبالتالي سنحصل على النتيجة التالية عند تنفيذ الكود أعلاه:

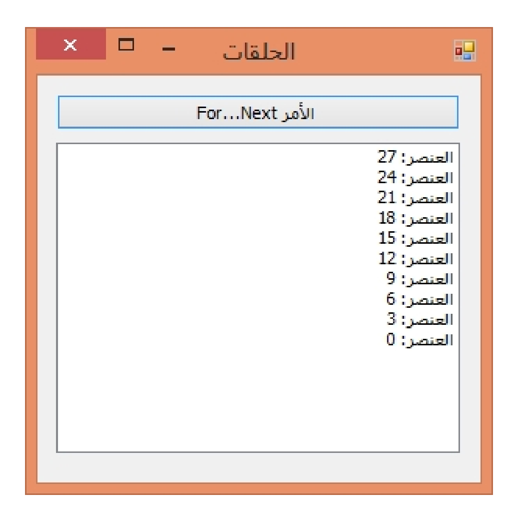


يمكننا كذلك جعل قيمة التدرج سلبية / تراجعية كما يلي:

```
Dim nCount As Integer

For nCount = 27 To 0 Step -3
    Me.lbNumbers.Items.Add("العنصر: " & nCount)
Next
```

بعد تنفيذ الكود أعلاه سنحصل على النتيجة الآتية:



الصيغة التكرارية الشرطية Do..Loop :

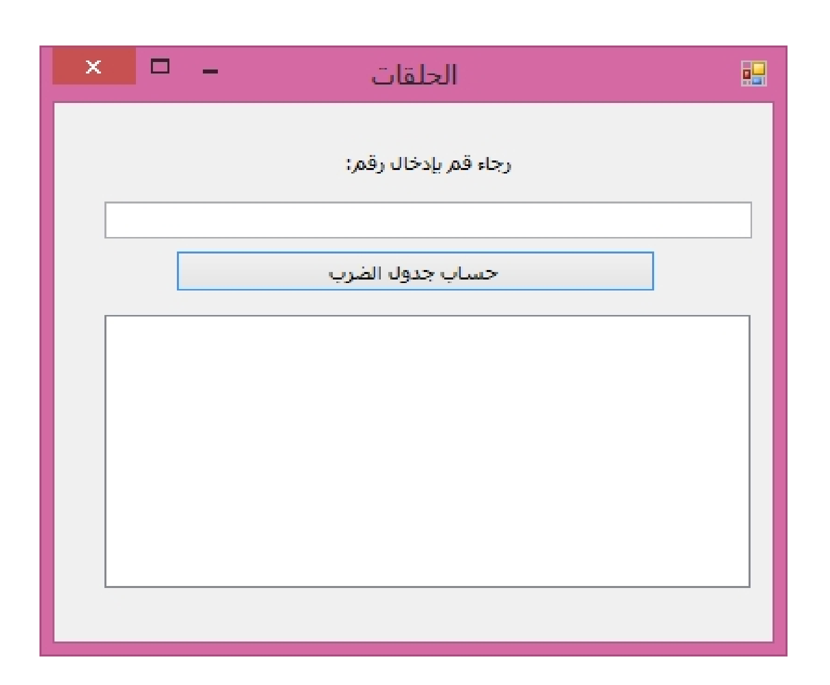
ترتكز البنية التكرارية من نوع Do..Loop على شرط التكرار وحسب نتيجته يتم إعادة تنفيذ الأوامر، ويوجد بها عدة أنواع سوف نتعرف عليها بالترتيب:

الصيغة التكرارية الشرطية Do..While :

هذه البنية التكرارية تعتمد على نتيجة الشرط الذي يأتي بعد الكلمة While، ويتم تكرار تنفيذ الأوامر مادامت نتيجة الشرط متحققة True، وهذه صيغتها:

```
Do
    'الأوامر
Loop While 'الشرط
```

سنتعرف الآن بحول الله على مثال لاستخدام هذه البنية التكرارية، قم بإنشاء مشروع جديد من نوع Windows Forms Application واسحبا على الفورم الأدوات التالية:



الأداة	اسمها	دورها
Label	lblNumber	من أجل عرض السؤال
TextBox	txtNumber	من أجل إدخال الرقم
Button	btnCalculate	من أجل حساب جدول ضرب الرقم
ListBox	lbDetails	من أجل عرض نتائج جدول الضرب

الآن أدخل إلى الحدث Click الخاص بالزر btnCalculate وقم بكتابة الكود الآتي:

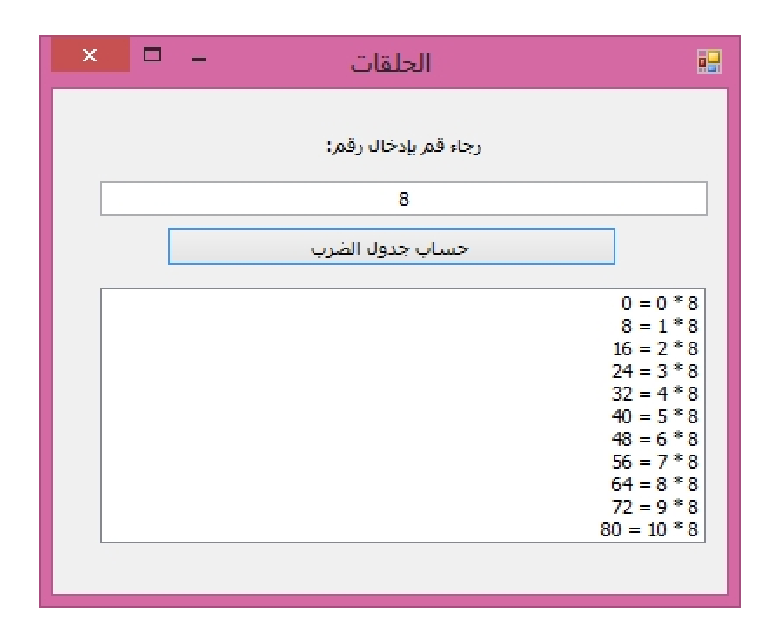
```
    Private Sub btnCheck_Click(sender As Object, e As
EventArgs) Handles btnCalculate.Click

        Dim Number As Integer = Val(txtNumber.Text)
        Dim Counter As Integer = 0
        Do
            lbDetails.Items.Add(Number & " * " & Counter & "
= " & Number * Counter)
            Counter += 1
        Loop While Counter <= 10

    End Sub
```

أعلنا في الأول عن متغير اسمه Number يستقبل القيمة المدخلة في مربع النص txtNumber بعد أن يقوم بتحويلها إلى قيمة رقمية من خلال الدالة Val ثم بعد ذلك أهلنا عن متغير عداد اسمه Counter أعطيناه القيمة 0 كقيمة بدئية، ثم بدأنا عملية التكرار التي تقوم بعرض نتيجة ضرب قيمة العدد المدخل Number مع قيمة العداد Counter التي تتغير تزايديا عند كل تكرار، ويتم تكرار هذه العملية التكرارية إلى أن تصبح قيمة العداد 10 أي أن جدول الضرب الذي سيعرض سيظهر لنا

نتيجة جداء العدد المدخل مع الأعداد الإحدى عشر الأولى بدء من 0 وانتهاء ب 10، أي بعد تنفيذ البرنامج لو أدخلنا القيمة 8 سنحصل على النتيجة الآتية:



يمكننا تقديم الكلمة While بعد الكلمة Do وسنحصل على نفس النتيجة كما يلي:

```
        Dim Number As Integer = Val(txtNumber.Text)
        Dim Counter As Integer = 0
        Do While Counter <= 10
            lbDetails.Items.Add(Number & " * " & Counter & "
= " & Number * Counter)
            Counter += 1
        Loop
```

الفرق بين تقديم While وبين تأخيرها يتمثل في أن الأولى تمنع تنفيذ الأمر إلا بعد أن يتحقق الشرط، بينما While التي تأتي في الأخير بعد Loop تسمح بتنفيذ الأمر للمرة الأولى حتى مع عدم تحقق الشرط، والكود التالي يوضح الفرق بين الأمرين.

تقديم While:

```
Do While 1 = 3
    MsgBox("هذا الكود لن ينفذ أبدا")
Loop
```

في الكود أعلاه لن يتم عرض الرسالة لأن شرط While غير متحقق لأن 1 لا يساوي ثلاثة ولأن While جاءت أولا في بداية التكرار.

تأخير While:

```
Do
    MsgBox("هذا الكود سينفذ مرة واحدة")
Loop While 1 = 3
```

في الكود أعلاه سيتم عرض الرسالة في كل مرة يتم تنفيذه لأن While أتت في نهاية التكرار على خلاف الكود الأول.

الصيغة التكرارية الشرطية Do..Until :

مثلها مثل البنية السابقة Do..While إلا أن هذه البنية تسمح بتكرار الأوامر إلى أن يتحقق الشرط وليس مادام الشرط متحققا، بمعنى أن التعليمات التي ستأتي بعد الكلمة Do سيتم تكرارها إلى غاية تحقق الشرط الذي يأتي بعد الكلمة Until.

الفرق بين Until و While:

Until	While
تسمح بتكرار الأوامر إلى أن يتحقق الشرط أي تصبح نتيجته True	تسمح بالتكرار مادام الشرط متحققا ويخرج من التكرار حينما لا يتحقق الشرط أي تصبح نتيجته False

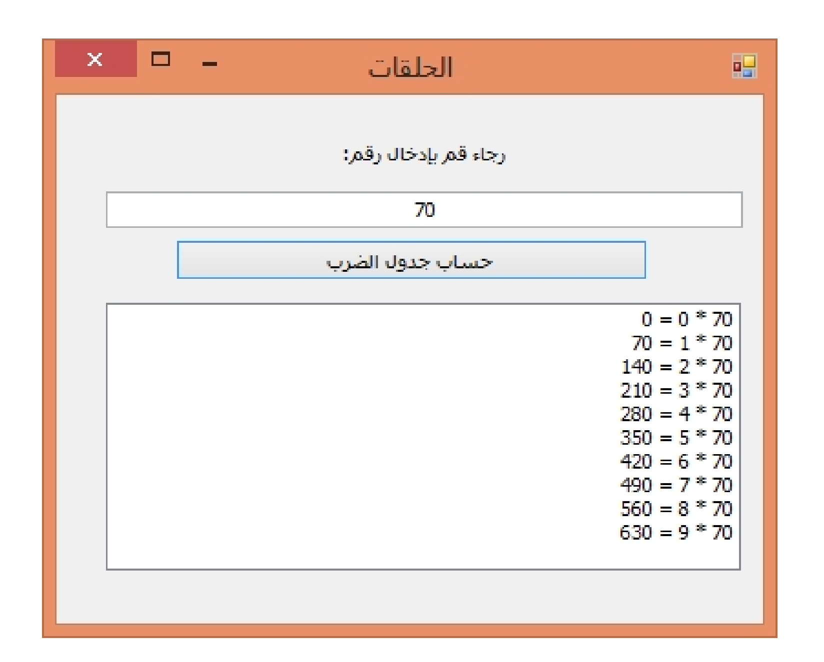
سنشتغل على نفس المثال السابق الذي يقوم بحساب جدول الضرب، لكن هذه المرة مع استخدام الأمر Until بدل While، أي أن الشفرة ستكون كما يلي:

```
    Private Sub btnCheck_Click(sender As Object, e As
EventArgs) Handles btnCalculate.Click

        Dim Number As Integer = Val(txtNumber.Text)
        Dim Counter As Integer = 0
        Do Until Counter = 10
            lbDetails.Items.Add(Number & " * " & Counter & "
= " & Number * Counter)
            Counter += 1
        Loop

    End Sub
```

لاحظ أننا غيرنا الشرط، بدل أن يكون العداد أصغر من أو يساوي 10 أصبح معنى الشرط، قم بتكرار الأوامر إلى أن يصبح العداد Counter مساويا ل 10 وفي كل مرة تزداد قيمته بواحد إلى أن يتحقق الشرط فينتهي التكرار، لو نفذنا الكود أعلاه وأدخلنا العدد 70 مثلا سوف نحصل على النتيجة التالية:



رأينا فيما تقدم أنه يمكننا تسبيق الكلمة While وتأخيرها، نفس الكلام ينطبق على الكلمة Until يمكننا وضعها في البداية بعد الكلمة Do ويمكننا كذلك وضعها في نهاية التكرار بعد الكلمة Loop، في الحالتين معا سيتم تنفيذ الأوامر مادام الشرط غير متحقق، بمعنى أن الكود التالي:

```
Do Until 1 = 3
    MsgBox("هذا الكود سينفذ من دون توقف لأن الشرط غير متحقق ولن يتحقق")
Loop
```

هو نفسه الكود التالي:

```
Do
    MsgBox("هذا الكود سينفذ من دون توقف لأن الشرط غير متحقق ولن يتحقق")
Loop Until 1 = 3
```

ففي الحالتين معا سيتم عرض الرسالة من دون توقف لأن الشرط غير متحقق دائما، ومادام الأمر كذلك فستستمر عملية تكرار الأوامر.

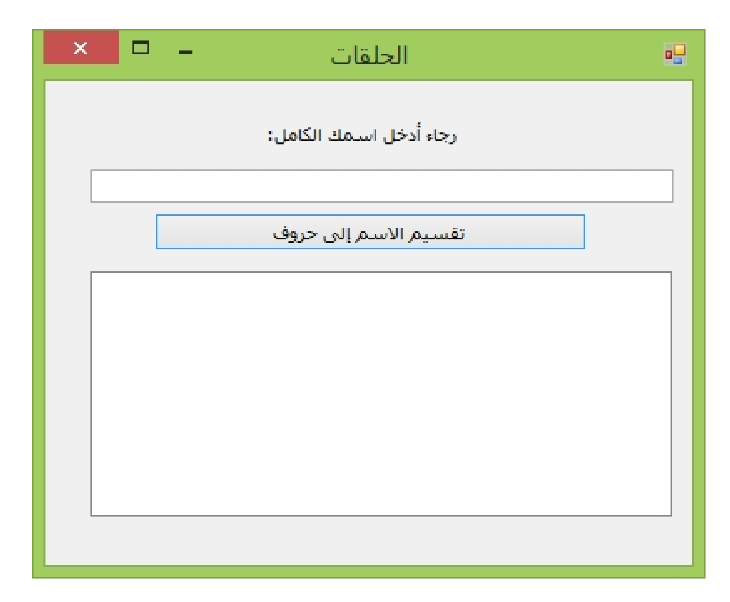
الصيغة التكرارية For Each...Next :

تسمح لنا هذه البنية التكرارية بتكرار أمر معين على جميع عناصر مجموعة معينة وصيغتها كما يلي:

```
For Each Item As DATA_TYPE In GROUP
    'الأوامر المراد تكرارها على العناصر
Next Item
```

سنقوم بإنشاء برنامج بسيط يستقبل اسم المستخدم ويقوم بتقسيمه وتوزيعه إلى حروف متفرقة وعرض كل حرف في سطر.

قم بإنشاء مشروع جديد من نوع Windows Forms Application وضع عليه الأدوات التالية:



الأداة	اسمها	دورها
Label	lblName	من أجل عرض السؤال
TextBox	txtName	من أجل إدخال الاسم
Button	btnSplit	من أجل تقسيم الاسم وتوزيعه
ListBox	lbDetails	من أجل عرض الحروف المكونة للاسم

انقر على الزر btnSplit مرتين من أجل الولوج إلى الحدث Click وقم بكتابة الكود التالي:

```
    Private Sub btnSplit_Click(sender As Object, e As
EventArgs) Handles btnSplit.Click

        Dim FullName As String = txtFullName.Text
```

```
        For Each chr As Char In FullName
            Me.lbDetails.Items.Add(chr)
        Next

    End Sub
```

في الشفرة أعلاه قمنا بالإعلان عن متغير اسمه FullName يستقبل القيمة النصية المدخلى في مربع النص txtFullName، ثم بعد ذلك بدأنا البنية التكرارية For Each التي تأخذ كل عنصر من عناصر المجموعة (في حالتنا هذه كل حرف من السلسلة النصية) وتقوم بعرضه في أداة ListBox.

الأمر With:

يسمح لنا الأمر With بتكرار مجموعة من الأوامر التي تشير إلى نفس الكائن Object على سبيل المثال، نريد تغيير خصائص زر معين بواسطة الكود، بدل أن نكتب الأوامر التالية:

```
        Button1.Text = "النص الظاهر"
        Button1.BackColor = Color.Blue
        Button1.Size = New Point(100, 40)
        Button1.ForeColor = Color. Yellow
        Button1.TextAlign = ContentAlignment.MiddleCenter
        Button1.Focus()
```

نقوم بتعيين اسم الكائن وفي حالتنا هذه هو الزر المسمى Button1 في الجزء With ثم نقوم باستدعاء الوظائف والخصائص مباشرة بعد كتابة رمز النقطة كما يلي:

```
        With Button1
            .Text = "النص  الظاهر"
```

```
        .BackColor = Color.Blue
        .Size = New Point(100, 40)
        .ForeColor = Color.Yellow
        .Focus()
        .TextAlign = ContentAlignment.MiddleCenter
    End With
```

المصفوفات Arrays

خلال الفقرات السابقة كنا نتعامل مع المتغيرات، ورأينا كيف تسمح لنا بتخزين القيم في الذاكرة، ورأينا كذلك أن كل متغير مخصص لاحتواء قيمة واحدة، ولا يمكن للمتغير الواحد أن يستقبل أكثر من قيمة في وقت واحد.

لذلك لو افترضنا أننا نريد إنشاء برنامج لحساب معدلات الطلبة، بحيث يستقبل هذا البرنامج نقطة الطالب في كل مادة من المواد التي يدرسها، ثم يقوم بحساب المعدل العام، قد تقول لي: بإمكاننا عمل ذلك من خلال الإعلان عن مجموعة من المتغيرات حسب عدد المواد لدينا ثم نجري عليها عملية حساب المعدل العام، سأقول لك بأن جوابك صائب، لأنه لو أردنا مثلا حساب معدل طالب معين في عشرة مواد يدرسها، سوف نعلن عن عشر متغيرات رقمية لاستقبال نقاط الطالب، بالطريقة التالية:

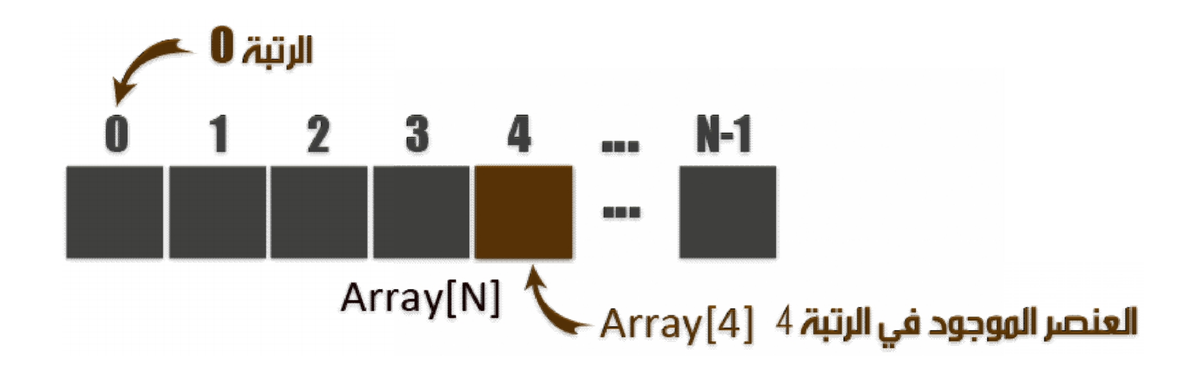
```
Dim Value1, Value2, Value3, Value4, Value5, Value6,
Value7, Value8, Value9, Value10 As Double
```

لكن هذه الطريقة ليست مجدية إذا كنا سنحتاج إلى تخزين قيم كثيرة، لأننا سنجد صعوبة في استحضار أسماء المتغيرات، ناهيك عن صعوبة تتبع الكود وقراءته.

لحل هذه المشاكل وغيرها وجد مفهوم المصفوفات Arrays، بحيث نستخدمها حينما نود تخزين مجموعة من القيم المنتمية لنفس نوع البيانات في متغير واحد، ويرتكز مفهوم المصفوفات على ركيزتين أساسيتين هما:

القيمة Value: وهي القيم المراد تخزينها في عناصر المصفوفة، لو أخذنا مثلا مصفوفة لتخزين درجات الطلاب في المواد فإن الدرجات هي القيم، كل درجة عبارة عن قيمة سيتم تخزينها في عنصر معين من عناصر مصفوفة المواد.

الرتبة Index: وهي رتبة العنصر داخل المصفوفة، وتبدأ بصفر وتنتهي برتبة آخر عنصر ناقص واحد، مثلا لو أردنا تخزين الدرجات في مصفوفة المواد فإن التمثيل الفعلي سيكون بالشكل التالي:



نستطيع الوصول إلى أي عنصر من عناصر المصفوفة من خلال رتبته Index، هذا النوع من المصفوفات الذي نتحدث عنه يسمى المصفوفات الأحادية البعد one-dimensional array لأنها تحتوي على بعد واحد يضم العناصر بشكل خطي كما يعرض الشكل أعلاه، بحيث يمكن تمثيلها على شكل خانات متسلسلة لكل خانة رتبة وقيمة.

الإعلان عن مصفوفة أحادية:

للإعلان عن مصفوفة أحادية في لغة الفيجوال بيسك، نستخدم نفس طريقة الإعلان عن المتغيرات مع إضافة قوسين لتحديد عدد العناصر الممكن تخزينها في هذه المصفوفة، في المثال التالي سنقوم بالإعلان عن مصفوفة رقمية من نوع عشري Double من أجل احتواء نقاط الطلاب:

```
Dim Marks(9) As Double
```

المصفوفة أعلاه مكونة من عشرة عناصر عشرية لأن الترتيب يبدأ من صفر وينتهي بآخر رتبة في المصفوفة.

إسناد القيم لعناصر المصفوفة الأحادية:

لإسناد القيم لعناصر المصفوفة، يمكننا ذلك في نفس وقت الإعلان كما يلي لكن لا ينبغي لنا تحديد عدد العناصر لأنه سيعرف من خلال القيم التي أسندناها لحظة الإعلان:

```
Dim Marks() As Double = {20, 19, 14, 17, 13, 11.5, 16.25, 18.75, 9, 15}
```

في المثال أعلاه أنشأنا مصفوفة رقمية مكونة من 10 عناصر، العنصر قيمهم كما يلي:

رتبة العنصر	قيمته
0	20
1	19
2	14
3	17
4	13
5	11.5
6	16.25
7	18.75
8	9
9	15

ويمكننا كذلك إسناد القيم بعد الإعلان عن المصفوفة وتحديد عدد عناصرها كما يوضح المثال التالي الذي يؤدي نفس الكود السابق:

```
Dim Marks(9) As Double

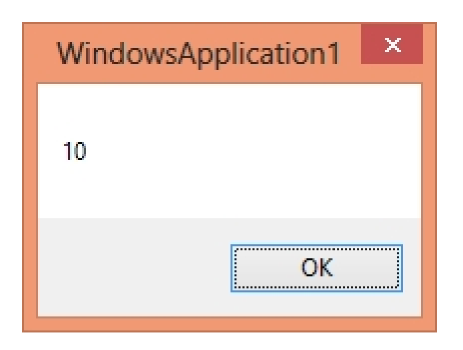
Marks(0) = 20
Marks(1) = 19
Marks(2) = 14
Marks(3) = 17
Marks(4) = 13
Marks(5) = 11.5
Marks(6) = 16.25
Marks(7) = 18.75
Marks(8) = 9
Marks(9) = 15
```

الحصول على طول المصفوفة:

لمعرفة عدد العناصر المكونة للمصفوفة يمكننا ذلك من خلال الخاصية Length كما يوضح المثال التالي:

```
        Dim Marks() As Double = {20, 19, 14, 17, 13, 11.5,
16.25, 18.75, 9, 15}
        MsgBox(Marks.Length)
```

بعد تنفيذ الكود أعلاه سنحصل على رسالة مفادها أن عدد العناصر المكونة للمصفوفة هو 10 عناصر كما تبين الصورة التالية:



و يمكننا أيضا الحصول على طول المصفوفة باستخدام الخاصية Count كما يبين المثال التالي:

```
        Dim Marks() As Double = {20, 19, 14, 17, 13, 11.5,
16.25, 18.75, 9, 15}

        MsgBox(Marks.Count)
```

سنحصل على نفس النتيجة السابقة.

معالجة عناصر المصفوفة:

لعرض عنصر من عناصر المصفوفة أو استخدامه في عملية معينة، نحدد رتبة العنصر الذي نريد، كما تعرض لنا الأمثلة التالية:

```
        Dim Marks() As Double = {20, 19, 14, 17, 13, 11.5, 16.25, 18.75, 9, 15}

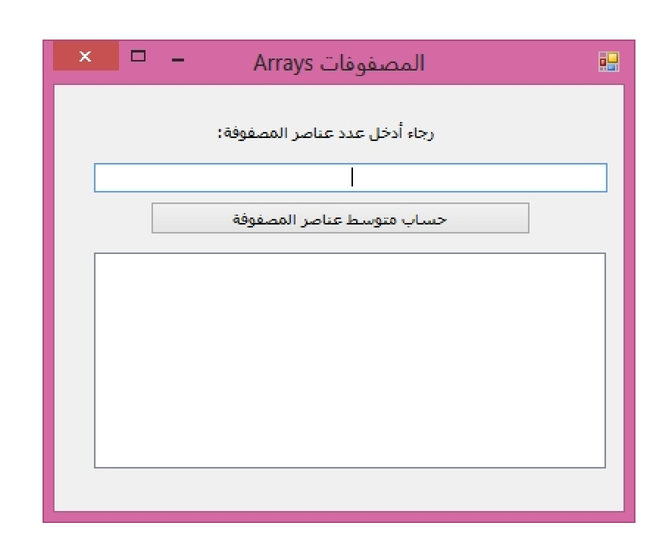
        Dim Value As Double = Marks(2)

        MsgBox(Marks(8))

        Dim Sum As Double = Marks(1) + Marks(4)
```

ولكي نتعرف أكثر على كيفية معالجة عناصر المصفوفات، سنقوم بتصميم برنامج بسيط يطلب من المستخدم أن يدخل عدد عناصر مصفوفة رقمية، ثم نقوم بإسناد قيم عشوائية لهذه العناصر ثم نحسب المتوسط Average ونعرضه في رسالة ونطبع قيم العناصر في أداة ListBox.

قم بإنشاء مشروع جديد من نوع Windows Forms Application وضع عليه الأدوات التالية:



الأداة	اسمها	دورها
Label	lblArray	من أجل عرض السؤال
TextBox	txtArray	من أجل إدخال عدد عناصر المصفوفة
Button	btnAvg	من أجل حساب متوسط المصفوفة
ListBox	lbDetails	من أجل عرض قيم عناصر المصفوفة

الآن انقر مرتين على الزر btnAvg من أجل الولوج إلى الحدث Click وقم بكتابة الكود التالي:

```
    Private Sub btnAvg_Click(sender As Object, e As
EventArgs) Handles btnAvg.Click

        Dim ItemCount As Integer = Val(txtArray.Text)

        Dim intArray(ItemCount) As Integer

        Dim Rand As New Random

        Dim Sum As Integer = 0

        For Count As Integer = 0 To intArray.Length - 1

            intArray(Count) = Rand.Next(100)

            lbDetails.Items.Add(intArray(Count))

            Sum += intArray(Count)
        Next

        Dim Average As Double = Sum / intArray.Length

        MsgBox("المتوسط الحسابي للمصفوفة هو: " & Average)

    End Sub
```

الآن تعال بنا نشرح هذا الكود أمرا بأمر:

```
Dim ItemCount As Integer = Val(txtArray.Text)
```

في المتغير ItemCount قمنا بتخزين عدد عناصر المصفوفة المدخل بواسطة المستخدم.

```
Dim intArray(ItemCount) As Integer
```

ثم أعلنا عن مصفوفة اسمها intArray مكونة من عدد العناصر المدخل من طرف المستخدم.

```
Dim Rand As New Random
```

الكائن Rand أنشأناه بغرض الحصول على قيم رقمية عشوائية حسب المجال الذس سوف نحدده له فيما بعد.

```
Dim Sum As Integer = 0
```

هذا المتغير Sum أنشأناه بغرض الحصول على مجموع قيم عناصر المصفوفة لكي نستخدمه أثناء حساب المتوسط.

```
For Count As Integer = 0 To intArray.Length - 1
```

هذا التكرار يبدأ من 0 وينتهي بآخر عنصر في المصفوفة.

```
intArray(Count) = Rand.Next(10)
```

في كل تكرار سنقوم بإسناد قيمة عشوائية جديدة أقل من 100 للعنصر الذي يصله التكرار من خلال الدالة Next التابعة للكائن Rand المستنسخ من الفئة Random.

```
lbDetails.Items.Add(intArray(Count))
```

ثم نقوم بعرض قيمة العنصر الذي يصله التكرار في أداة ListBox التي أسميناها lbDetails.

```
Sum += intArray(Count)
```

في كل مرة نقوم بإضافة قيمة العنصر إلى مجموع قيم العناصر التي مر عليها التكرار.

ثم بعد ذلك نخرج من التكرار ونقوم بحساب المتوسط وصيغته كما هو معلوم هي: مجموع قيم العناصر مقسوم على عدد العناصر كما يوضح الأمر التالي:

```
Dim Average As Double = Sum / intArray.Length
```

بعد ذلك قمنا بطباعة النتيجة في رسالة.

```
MsgBox("المتوسط الحسابي للمصفوفة هو: " & Average)
```

لو نفذنا الكود أعلاه كاملا وأدخلنا مثلا القيمة 10 كعدد لعناصر المصفوفة فسوف تكون النتيجة كما يلي (مع الأخذ بعين الاعتبار أن قيم العناصر ستكون عشوائية):

المصفوفات Arrays

رجاء أدخل عدد عناصر المصفوفة:

10

حساب متوسط عناصر المصفوفة

1
2
9
1
8
4
6
0
4
0
0

WindowsApplication1

المتوسط الحسابي للمصفوفة هو: 3.18181818181818

OK

مثال آخر:

سنقوم الآن بالتعرف على مثال يوضح كيفية استخدام المصفوفات الأحادية البعد، وهذه المرة سنتعامل مع المصفوفات النصية.

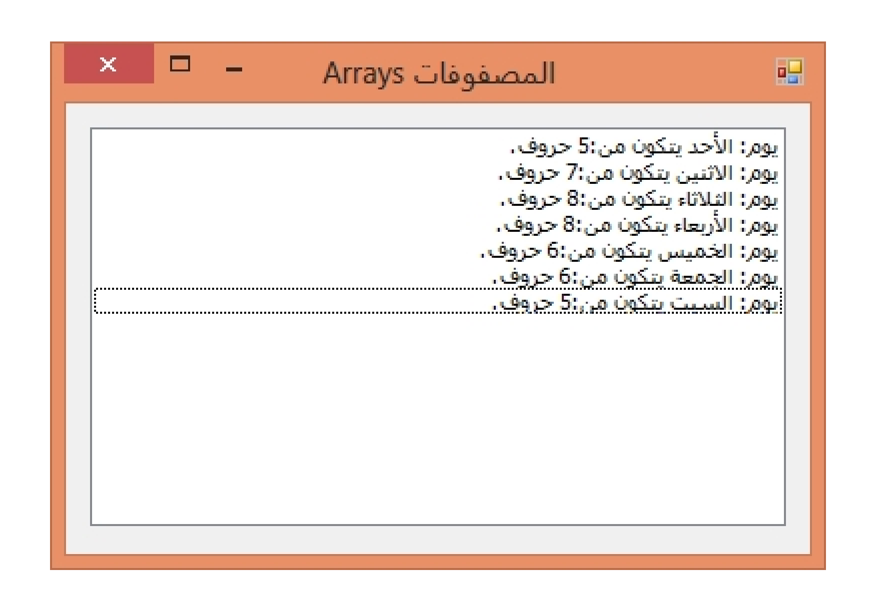
البرنامج الذي سنقوم بإنشائه الآن هو برنامج بسيط يقوم بعرض قيمة كل عنصر مصحوبة بعدد الحروف المكونة له:

```
Dim WeekArray(6) As String

WeekArray(0) = "الأحد"
WeekArray(1) = "الاثنين"
WeekArray(2) = "الثلاثاء"
WeekArray(3) = "الأربعاء"
WeekArray(4) = "الخميس"
WeekArray(5) = "الجمعة"
WeekArray(6) = "السبت"

For Each Day As String In WeekArray
    lbDetails.Items.Add("يوم: " & Day & " يتكون من:" & 
Day.Length & " حروف. ")
Next
```

بعد تنفيذ هذا الكود ستكون النتيجة هكذا:



ترتيب المصفوفات Sorting Arrays:

يمكننا ترتيب المصفوفات النصية حسب الترتيب الأبجدي، كما يمكننا ترتيب المصفوفات الرقمية من الأصغر إلى الأكبر، لعمل ذلك تتيح لنا لغة الفيجوال بيسك الدالة Sort التابعة للفئة Array، ويمكننا استخدامها كما يلي:

```
Dim NamesArray(4) As String

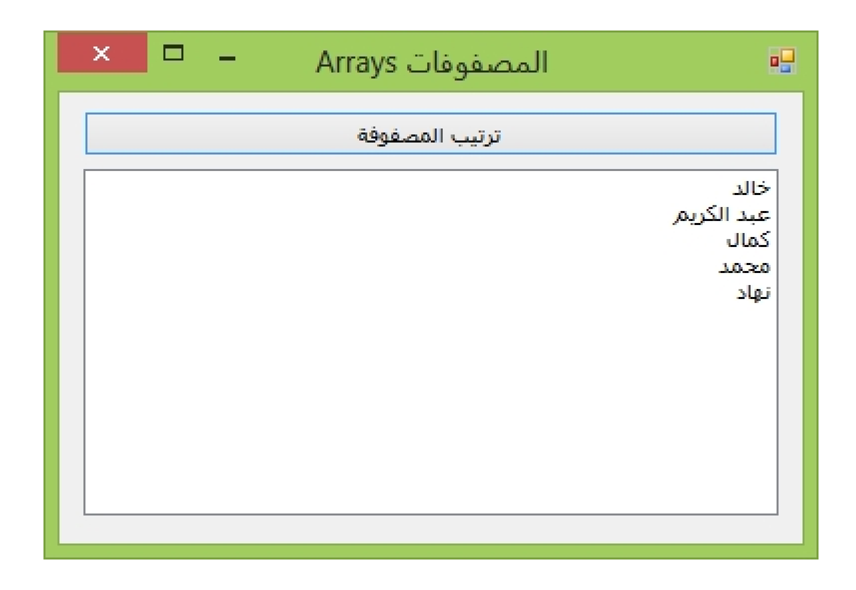
NamesArray(0) = "خالد"
NamesArray(1) = "نهاد"
NamesArray(2) = "محمد"
NamesArray(3) = "كمال"
NamesArray(4) = "عبد الكريم"

Array.Sort(NamesArray)

Me.lbDetails.Items.AddRange(NamesArray)
```

الدالة AddRange تسمح بإضافة جميع عناصر المصفوفة إلى أداة ListBox بدل استخدام الآليات التكرارية أو إضافة كل عنصر على حدة من خلال الدالة Add.

عند تنفيذ الكود أعلاه سوف نحصل على النتيجة التالية:



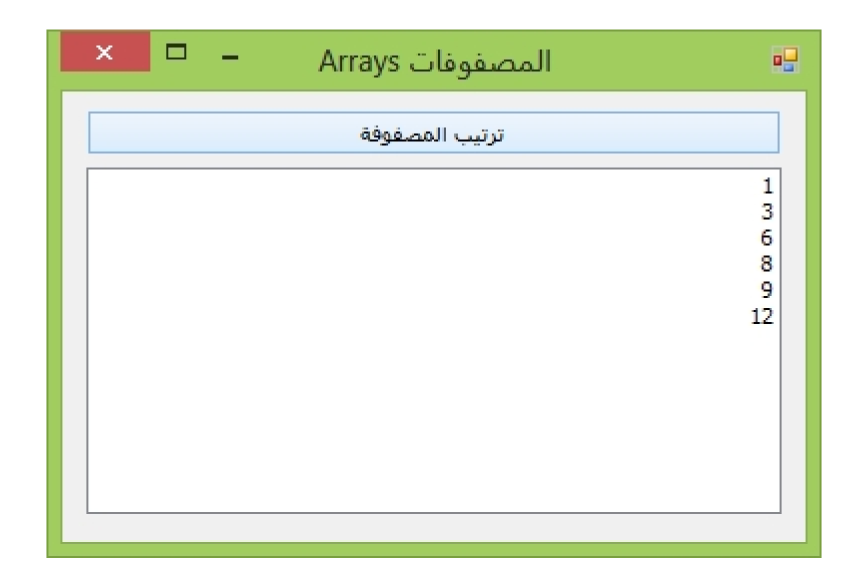
لاحظ أن الأسماء تم ترتيبها حسب الحروف الأبجدية، نفس الكلام ينطبق على الأرقام، ولكي نفهم الأمر أكثر شاهد المثال التالي:

```
Dim NumbersArray() As Integer = {8, 6, 1, 9, 12, 3}

Array.Sort(NumbersArray)

For Count As Integer = 0 To NumbersArray.Count - 1
    Me.lbDetails.Items.Add(NumbersArray(Count))
Next
```

عند تنفيذ الكود أعلاه سوف نحصل على عناصر المصفوفة مرتبة بشكل تزايدي (من الأصغر إلى الأكبر):



قلب عناصر المصفوفة

تعرفنا قبل قليل على ترتيب المصفوفات، وذكرنا أنه يمكننا ترتيب عناصر المصفوفات تزايديا من خلال الدالة Sort التابعة للفئة Array، الآن سوف نتعرف على عملية قلب عناصر المصفوفة ليصبح آخر عنصر هو أو عنصر وهكذا دواليك من خلال استخدام الدالة Reverse التابعة لنفس الفئة Array كما يعرض لنا المثال التالي:

```
Dim myArray() As Integer = {4, 2, 3, 1}

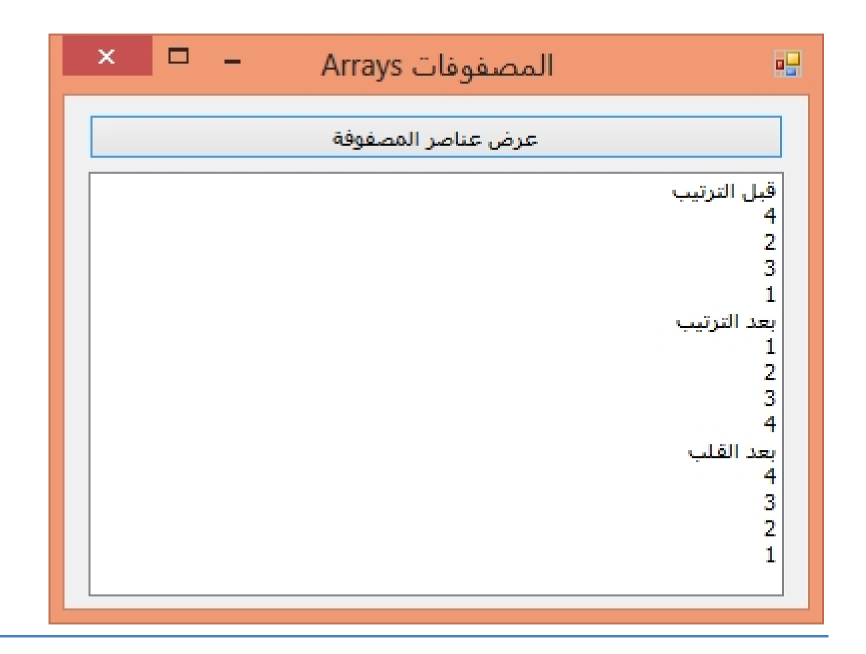
'عرض المصفوفة قبل الترتيب
Me.lbDetails.Items.Add("قبل الترتيب")
For Each Item As Integer In myArray
    Me.lbDetails.Items.Add(Item)
Next

'عرض المصفوفة بعد الترتيب
Array.Sort(myArray)
Me.lbDetails.Items.Add("بعد الترتيب")
For Each Item As Integer In myArray
    Me.lbDetails.Items.Add(Item)
```

```
Next

'عرض المصفوفة بعد القلب
Array.Reverse(myArray)
Me.lbDetails.Items.Add("بعد القلب")
For Each Item As Integer In myArray
    Me.lbDetails.Items.Add(Item)
Next
```

عند تنفيذ الكود أعلاه سوف نحصل على النتيجة الآتية:



المصفوفات المتعددة الأبعاد

تمثل المصفوفات المتعددة الأبعاد نوعا آخر من المصفوفات النادرة الاستخدام وأشهرها هي المصفوفات الثنائية البعد والتي يمكننا تمثيلها على شكل جدول يتكون من أسطر وأعمدة، ويتم الإعلان عنها بالصيغة التالية:

```
Dim TwoDimArray(2, 5) As Integer
```

ويمكننا أيضا الإعلان عنها كما يلي:

```
Dim TwoDimArray(0 To 2, 0 To 5) As Integer
```

المصفوفة الثنائية التي أعلنا عنها أعلاه مكونة من 3 أسطر و 6 أعمدة، أي أنها تحتوي على 6 * 3 عنصرا أي 18 عنصرا.

يمكننا تمثيل عناصر هذه المصفوفة الثنائية على شكل جدول كما يلي:

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)

يمكننا اسناد القيم لهاته العناصر من خلال رتبها، فيما يلي أمثلة على ذلك:

```
Dim TwoDimArray(2, 5) As Integer

TwoDimArray(0, 3) = 23

TwoDimArray(1, 4) = 195

TwoDimArray(2, 1) = 56
```

إذا أردنا أن نعالج عناصر المصفوفات الثنائية البعد من خلال تكرار، علينا القيام بتكرارين، الأول يذهب من أول سطر إلى آخر سطر، والتكرار الثاني يذهب من أول عمود إلى آخر عمود، فيما يلي مثال بسيط نستعرض من خلاله مصفوفة مكونة من سطرين وثلاثة أعمدة أي أنها تحتوي على 6 عناصر، ثم سنقوم بإسناد بعض القيم لهذه العناصر بعد ذلك سنقوم بعرضها:

```
'الإعلان عن المصفوفة الثنائية
Dim myArray(1, 2) As Integer

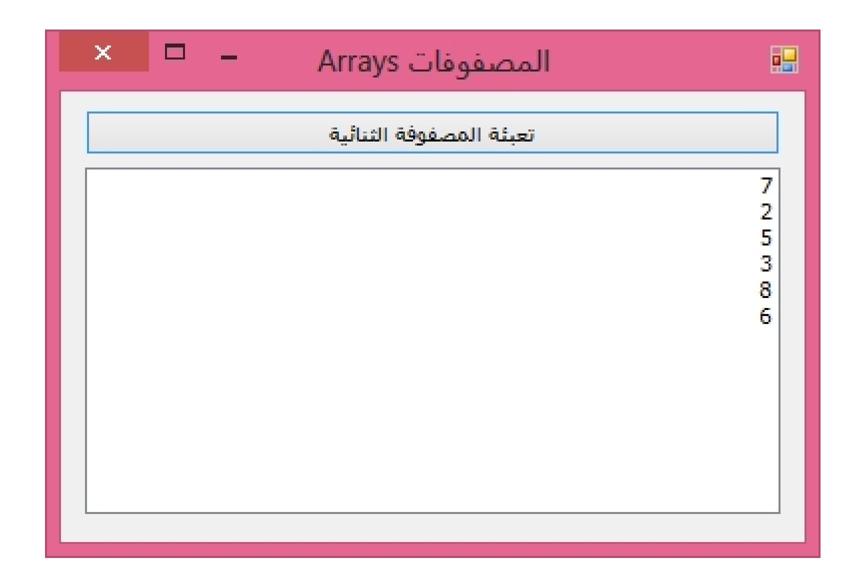
'تعبئة السطر الأول
myArray(0, 0) = 7
myArray(0, 1) = 2
myArray(0, 2) = 5

'تعبئة السطر الثاني
myArray(1, 0) = 3
myArray(1, 1) = 8
myArray(1, 2) = 6

'الإعلان عن عداد للأسطر وللأعمدة
Dim row, col As Integer

'التكرار الأول من أجل الأسطر
For row = 0 To 1
    'التكرار الثاني من أجل الأعمدة
    For col = 0 To 2
        lbDetails.Items.Add(myArray(row, col))
    Next
Next
```

أعلنا عن مصفوفة اسمها myArray مكونة من سطرين وثلاثة أعمدة ووضعنا بها مجموعة من القيم، ثم قمنا بعرض هاته القيم من خلال البنيات التكرارية، عند تنفيذ هذا الكود سوف نحصل على النتيجة التالية:

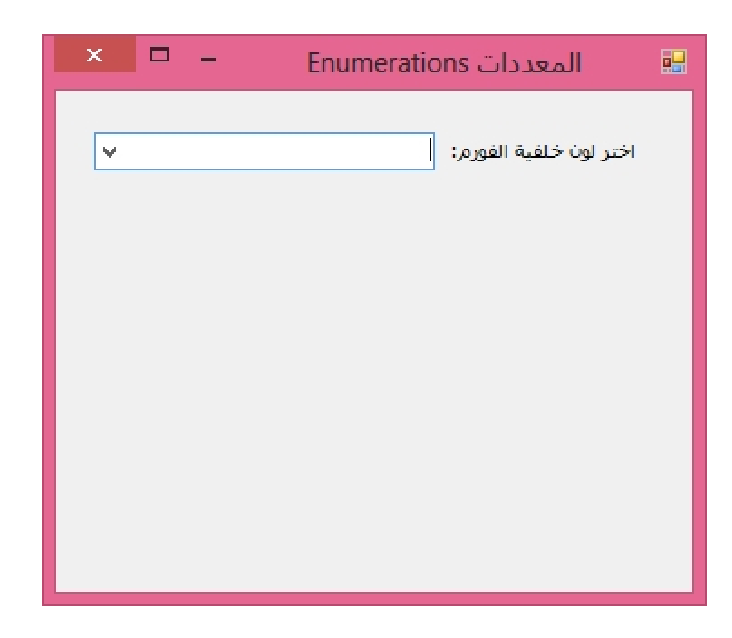


المعددات Enumerations:

نستخدم المعددات إذا كنا نريد تحديد قيم محددة ثابتة ليتم تخزينها في متغير ما، مثلا حينما نريد حفظ أيام الأسبوع في برنامجنا، فنحن نعلم مسبقا أن مجال أيام الأسبوع محدد والقيم معروفة لذلك يمكننا استخدام المعددات بدل الإعلان عن مجموعة من الثوابت Constants.

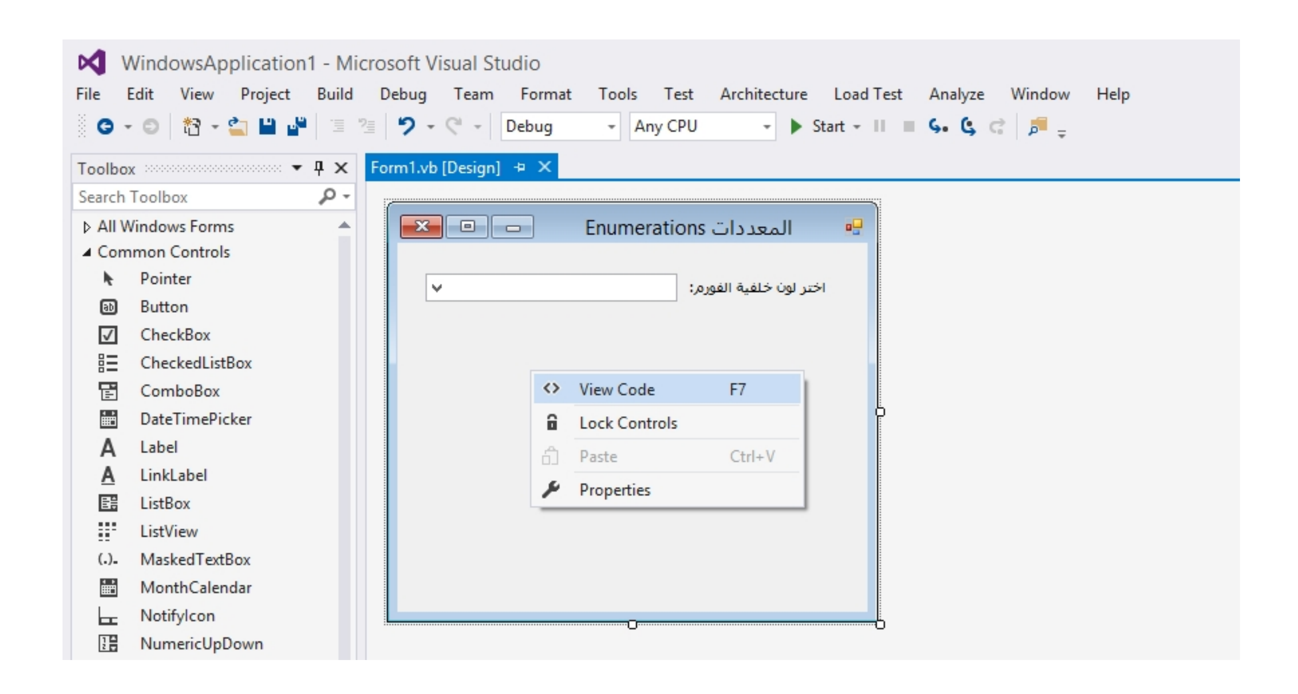
حينما يكون لدينا مجموعة من الثواب التي تنتمي إلى نفس المجال، مثلا أيام أسبوع، أسماء فواكه، ألوان... فيفضل أن نقوم بتجميعها داخل المعددات ليصبح استخدامنا لهذه الأنواع مرنا بدل تشتيت الكود في الإعلان عن مجموعة من الثوابت.

قم بإنشاء مشروع جديد من نوع Windows Forms Application وقم بتصميم الواجهة كما يلي:



الأداة	اسمها	دورها
Label	lblColor	من أجل عرض النص
ComboBox	cmbColor	من أجل اختيار لون خلفية الفورم

الآن سنقوم بإنشاء معددة Enumerations خاصة بأسماء الألوان التي نريد تعبئة الكومبوبوكس بها، قم بالضغط بيمين الماوس على الفورم واختر View Code كما يلي:



بعد ذلك قم بإنشاء المعددة كما يلي:

```
Public Class Form1

    Enum EnumColors
        Red
        Blue
        Green
        Purple
        Orange
    End Enum

End Class
```

المعددة EnumColors تحتوي على مجموعة من الثوابت التي لها رتب تبدأ من صفر وتنتهي برتبة آخر عنصر ناقص واحد، كما يلي:

الثابت	قيمته
Red	0
Blue	1
Green	2
Purple	3
Orange	4

يمكننا التحكم في رتب العناصر المكونة للمعددة عبر إسناد القيمة الرقمية لكل ثابت من الثوابت كما يلي:

```
Enum EnumColors
    Red = 4
    Blue = 2
    Green = 1
    Purple = 3
    Orange = 0
End Enum
```

عند التعامل مع عناصر هذه المعددة يبكون الترتيب مبنيا على القيم الرقمية التي تم إسنادها لكل عنصر.

الآن سنقوم بتعبئة الكومبوبوكس الذي أسميناه cmbColor بأسماء الألوان إما يدويا أو من خلال إنشاء مصفوفة، قم بالولوج إلى الحدث Form_Load الخاص بالفورم من خلال النقر عليه مرتين واكتب الشفرة الآتية:

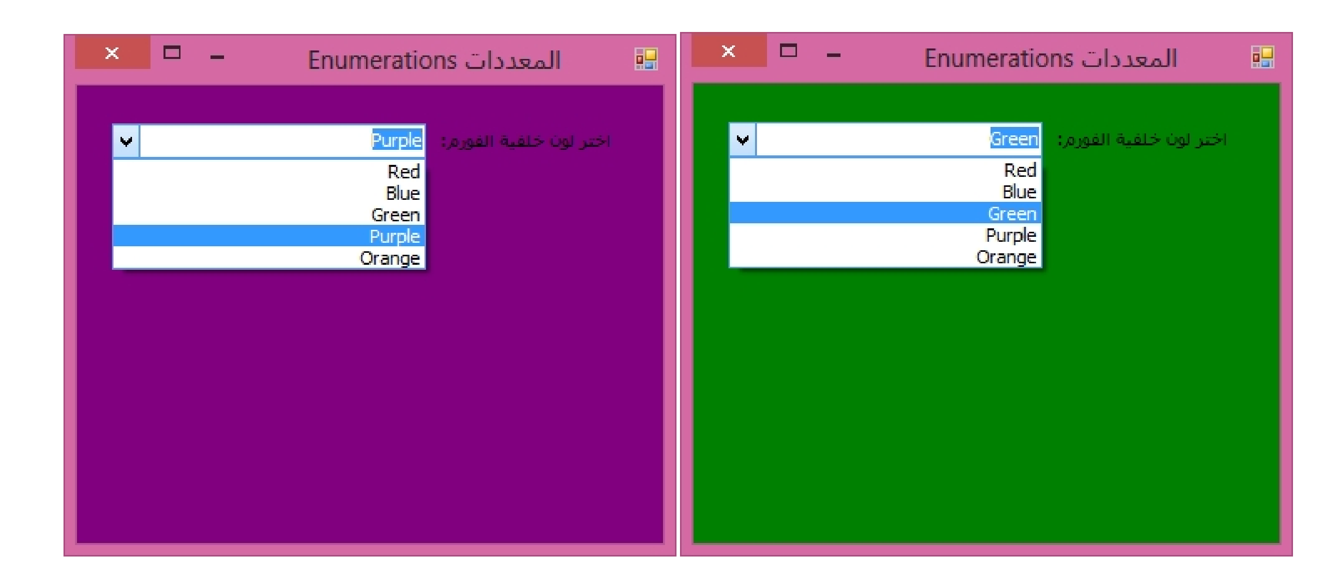
```
Dim ArrayColors() As String = {"Red",
                               "Blue",
                               "Green",
                               "Purple",
                               "Orange"}
```

```
Me.cmbColor.Items.AddRange(ArrayColors)
```

بعد ذلك، قم بالنقر مرتين على أداة ComboBox التي أسميناها cmbColor من أجل الانتقال إلى الحدث SelectedIndexChanged الذي يصدر عند اختيار عنصر من عناصر الكومبوبوكس، ثم اكتب الشفرة التالية:

```
Select Case cmbColor.SelectedIndex
    Case EnumColors.Red
        Me.BackColor = Color.Red
    Case EnumColors.Blue
        Me.BackColor = Color.Blue
    Case EnumColors.Green
        Me.BackColor = Color.Green
    Case EnumColors.Purple
        Me.BackColor = Color.Purple
    Case EnumColors.Orange
        Me.BackColor = Color.Orange
End Select
```

الكود أعلاه يتحقق من رتبة القيمة المحددة في الكومبوبوكس، فإن كانت تساوي صفر وهي نفس رتبة اللون الأحمر في المعددة EnumColors يتم تغيير لون خلفية الفورم إلى الأحمر، وهكذا دواليك مع باقي الألوان، عند تنفيذ الكود أعلاه سوف نحصل على النتيجة التالية:



فيما يلي الكود الكامل لهذا البرنامج البسيط الذي استعرضنا فيه مفهوم المعدات Enumerations:

```
Public Class Form1

    Enum EnumColors
        Red
        Blue
        Green
        Purple
        Orange
    End Enum

    Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load

        Dim ArrayColors() As String = {"Red",
                                       "Blue",
                                       "Green",
                                       "Purple",
                                       "Orange"}

        Me.cmbColor.Items.AddRange(ArrayColors)

    End Sub
```

```
    Private Sub cmbColor_SelectedIndexChanged(sender As
Object, e As EventArgs) Handles cmbColor.SelectedIndexChanged
        Select Case cmbColor.SelectedIndex
            Case EnumColors.Red
                Me.BackColor = Color.Red
            Case EnumColors.Blue
                Me.BackColor = Color.Blue
            Case EnumColors.Green
                Me.BackColor = Color.Green
            Case EnumColors.Purple
                Me.BackColor = Color.Purple
            Case EnumColors.Orange
                Me.BackColor = Color.Orange
        End Select
    End Sub

End Class
```

معالجة النصوص Handling Strings

توفر لنا لغة الفيجوال بيسك مجموعة من الدوال التي تمكننا من التعامل مع النصوص وإجراء مختلف العمليات التي ترغب فيها من تقطيع ودمج وبحث وتصغير وتكبير حالات الأحرف وغير ذلك من الدوال المهمة التي سنستعرضها بالترتيب مستعرضين لكل دالة مثالا تطبيقيا.

الحصول على طول النص:

إذا أردنا أن نعرف عدد الحروف المكونة لنص معين، يمكننا استخدام الخاصية Length وكذلك الدالة Len وفيما يلي كيفية استخدامهما:

```
Dim Name As String = "Khalid"
MsgBox(Name.Length)
'أو
MsgBox(Len(Name))
```

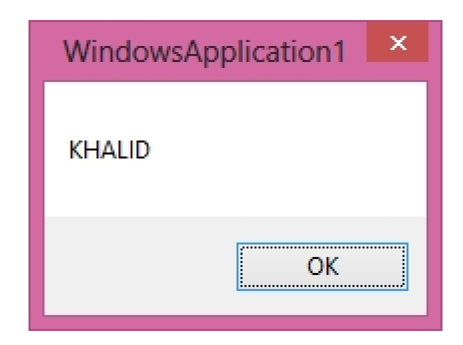
في الحالتين معا، سيتم عرض رسالة تحتوي على طول السلسلة النصية وهو 6 أحرف.

تكبير حالة الأحرف:

إذا أردنا تكبير حالة الأحرف، فيمكننا عمل ذلك بواسطة الدالة UCase، وكذلك الخاصية ToUpper كما يوضح المثال التالي:

```
Dim Name As String = "khalid"
MsgBox(Name.ToUpper())
'أو
MsgBox(UCase(Name))
```

النتيجة الحاصلة من الدالة UCase أو الخاصية ToUpper هي نفس القيمة النصية khalid لكن بحالة أحرف كبيرة هكذا KHALID كما توضح هذه الرسالة:

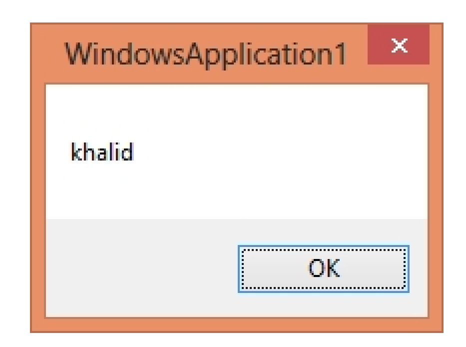


تصغير حالة الأحرف

بمقابل الدالة UCase التي تقوم بتكبير حالة الأحرف، يوجد لدينا الدالة LCase التي تقوم بتصغير حالة الأحرف، وكذلك بمقابل الخاصية ToUpper تتيح لنا لغة الفيجوال بيسك الخاصية ToLower التي تسمح بتصغير حالة الحروف، كما يوضح لنا المثال التالي:

```
Dim Name As String = "KHALID"
MsgBox(Name.ToLower())
'أو
MsgBox(LCase(Name))
```

سنحصل على النتيجة التالية في الحالتين معا:



اجتزاء النصوص SubString

توفر لنا لغة الفيجوال بيسك دالتين للحصول على جزء معين من سلسلة نصية، الدالة الأولى هي Mid والدالة Substring التابعة للفئة String، الدالة الأولى Mid تنتظر ثلاثة برامترات كما يلي:

```
Mid(string_Name, start_Index, Length)
```

البرامتر الأول string_Name نضع فيه النص الأصلي الذي نريد اجتزاءه.

البرامتر الثاني start_Index نحدد فيه رتبة أول حرف نريد أن يبدأ منه النص الجديد الناتج عن الاجتزاء علما أن رتبة أول حرف في النص الأصلي هي 1 وليست 0.

البرامتر الثالث Length نضع فيه عدد النص الذي نريد الحصول عليه.

وهذه أمثلة على كيفية استخدام الدالة Mid من أجل اجتزاء النصوص:

```
Dim Name As String = "KHALID"
Dim strResult1 As String = Mid(Name, 1, 3)
Dim strResult2 As String = Mid(Name, 2, 4)
```

في المتغير الأول strResult1 قمنا بتخزين القيمة المجتزأة من النص الأصلي KHALID بدء من أول حرف وطول نتيجة النص الجديد هي 3، أي أن النتيجة هي KHA.
بينما في المتغير الثاني strResult2 قمنا بتخزين القيمة المجتزأة من النص الأصلي KHALID بدء من ثاني حرف وطول نتيجة النص الجديد هو 4، أي أن النتيجة هي HALI.

الدالة الثانية التي تمكننا من تجزئة النص هي SubString التابعة للفئة String، وصيغتها كما يلي:

```
String_Name.Substring(start_Index, Length)
```

نستبدل الكلمة String_Name باسم النص الذي نريد تقطيعه، ثم نحدد في البرامتر Start_Index بداية التقطيع التي تبدأ من 0 على خلاف الدالة MID التي تعتبر أن أول حرف في السلسلة النصية رتبته 1. بعد ذلك نحدد طول النص المراد الحصول عليه بعد عملية الاجتزاء، كما توضح الأمثلة التالية:

```
Dim Name As String = "KHALID"

Dim strResult1 As String = Name.Substring(0, 3)

Dim strResult2 As String = Name.Substring(2, 4)
```

المتغير الأول سيحتوي على نتيجة النص بعد التقطيع الذي يبدأ من أول حرف و يحسب ثلاثة حروف معه أي أن strResult1 سيحتوي على KHA.

أما المتغير الثاني فإن قيمته ستكون هي نتيجة التقطيع بدءا من الحرف الثالث (لأن العد في الدالة SubString يبدأ من 0) وطول النص لدينا هو 4 وبالتالي سيحتوي المتغير strResult2 على ALID.

تقسيم النصوص Splitting

تمكننا الدالة Split التابعة للفئة String من تقطيع النص إلى مجموعة من العناصر وتخزينها في مصفوفة إذ سيتم اعتبار كل جزء من أجزاء النص عنصرا من عناصر المصفوفة الناتجة.

في المثال التالي، سنقوم بتخزين مجموعة من الأسماء المفصولة بعلامة شرطة في متغير نصي، ثم نقوم بتقطيع هذا النص حسب هذه الشرطة بواسطة الدالة Split ثم نخزن القيم الناتجة في مصفوفة نصية:

```
Dim Names As String = "Khalid-Hamid-Nihad-Hussain"

Dim ArrayNames() As String = Names.Split("-")

MsgBox(ArrayNames(0)) 'display Khalid
MsgBox(ArrayNames(1)) 'display Hamid
MsgBox(ArrayNames(2)) 'display Nihad
MsgBox(ArrayNames(3)) 'display Hussain
```

كما تلاحظ معي فالدالة Split تستلزم منا تحديد الرمز المراد اعتماده في التقطيع، وفي حالتنا هذه هو الشرطة-.

دمج النصوص Concatening

تسمح لنا لغة الفيجوال بيسك باستخدام مجموعة من الطرق لدمج النصوص، رأينا فيما تقدم معنا من فصول أولى أن الرمز & والرمز + يستخدمان لدمج القيم النصية، كما يبين لنا المثال التالي:

```
Dim First_Name As String = "Khalid"
Dim Last_Name As String = " ESSAADANI"

Dim FullName As String

FullName = First_Name + Last_Name
'Or
FullName = First_Name & Last_Name
```

المتغير FullName في الحالتين معا سيحتوي على نتيجة دمج قيمتي المتغيرين First_Name و Last_Name أي أن النتيجة ستكون: Khalid ESSAADANI.

توجد لدينا كذلك الدالة Concat التابعة للفئة String التي تقوم بدمج النصين الممررين لها كما يوضح المثال التالي:

```
Dim First_Name As String = "Khalid"
Dim Last_Name As String = " ESSAADANI"

Dim FullName As String

FullName = String.Concat(First_Name, Last_Name)
```

المتغير FullName سيضم قيمتي المتغيرين First_Name و Last_Name بعد أن تقوم الدالة Concat بدمجهما.

يوجد لدينا في لغة الفيجوال بيسك، الدالة Join التي تقوم بنفس الدور، لكن تتطلب منا تحديد رمز الفصل بين النصوص المراد دمجها، وهذا مثال على كيفية استخدام هذه الدالة:

```
Dim First_Name As String = "Khalid"
Dim Last_Name As String = "ESSAADANI"

Dim FullName As String

FullName = String.Join(" ", First_Name, Last_Name)
```

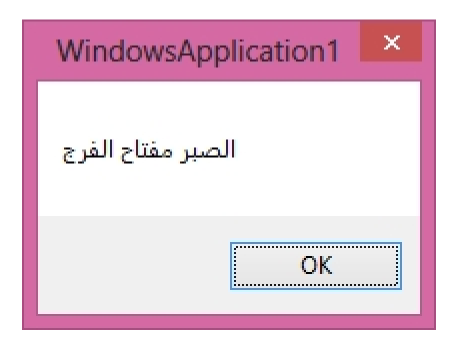
لاحظ أننا وضعنا في رمز الربط بين القيمتين رمز المساحة الفارغة Space لكي يتم ترك فراغ بين الاسم والنسب، ويمكننا الاستغناء عن تحديد رمز الربط وترك قيمته فارغة كما يمكننا وضع أي نص نريد وضعه كرابط للقيم المراد دمجها وهذا مثال أوضح على استخدام الدالة Join التابعة للفئة String:

```
Dim Expression1 As String = "الصبر"
Dim Expression2 As String = "الفرج"

Dim Expression3 As String = String.Join(" مفتاح ",
Expression1, Expression2)

MsgBox(Expression3)
```

بعد تنفيذ الكود أعلاه، سوف تحصل على النتيجة التالية:



البحث داخل النصوص Contains

نستطيع البحث عن قيمة معينة داخل قيمة نصية من خلال الدالة Contains التي تعيد لنا القيمة True في حال العثور على القيمة المبحوث عنها، وتعيد لنا القيمة False في حال عدم العثور عليها، وهذا مثال على كيفية استخدام الدالة Contains التابعة للفئة String:

```
Dim strText As String = "إذا فشلت في التخطيط فقد خططت للفشل"

If strText.Contains("التخطيط") = True Then
    MsgBox("القيمة المبحوث عنها موجودة")
Else
    MsgBox("آسف، القيمة المبحوث عنها غير موجودة")
End If
```

ستبحث الدالة Contains عن الكلمة "التخطيط" داخل المتغير النصي strText وتظهر لنا الرسالة حسب نتيجة البحث، في حالتنا هذه القيمة المعادة للدالة Contains هي True إذن ستظهر الرسالة الأولى: القيمة المبحوث عنها موجودة.

يمكننا القيام بنفس عملية البحث عن نص معين داخل سلسلة نصية، من خلال الدالة IndexOf التي تعيد لنا رتبة أول حرف من القيمة المبحوث عنها في حال العثور عليها، أما في حال عدم العثور عليها سوف تعيد لنا الدالة IndexOf القيمة السلبية-1 .

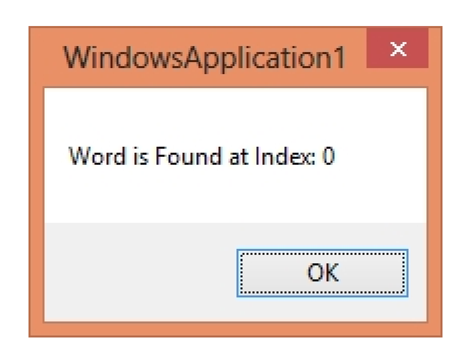
فيما يلي مثال يعرض للحالتين معا، الحالة الأولى تم العثور على النص المبحوث عنه، والحالة الثانية لم يتم العثور عليه.

الحالة الأولى:

```
Dim strText As String = "Think First, Code Later"

If strText.IndexOf("Think") > -1 Then
    MsgBox("Word is Found at Index: " &
           strText.IndexOf("Think"))
Else
    MsgBox("Not Found :(")
End If
```

الكلمة Think موجودة داخل قيمة المتغير strText إذن الدالة IndexOf سوف تعيد قيمة مخالفة ل-1، وبالتالي سيتم عرض الرسالة الأولى التي تفيد أن الكلمة المبحوث عنها تم العثور عليها مع تحديد رتبة أول حرف منها، كما تعرض الرسالة التالية:



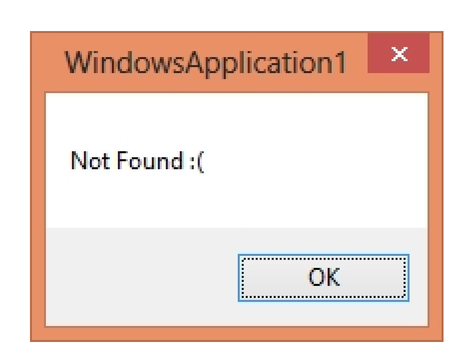
لأن الكلمة Think هي أول كلمة في السلسلة النصية التي شملتها عملية البحث فتم إرجاع الرتبة صفر لأن أول حرف من الكلمة يوجد في هذه الرتبة.

بينما في الحالة الثانية التي يعرضها الكود التالي فإن الدالة IndexOf لن تجد الكلمة المبحوث عنها وسوف تعيد لنا القيمة-1 وبالتالي ستظهر الرسالة التي تفيد أن الكلمة المبحوث عنها غير موجودة:

```
        Dim strText As String = "Think First, Code Later"

        If strText.IndexOf("Manage") = -1 Then
            MsgBox("Not Found :(")
        Else
            MsgBox("Word is Found at Index: " &
strText.IndexOf("Think"))
        End If
```

طبعا لأن الكلمة Manage غير موجودة فإن الشرط متحقق لأن الدالة IndexOf سوف تعيد بالفعل القيمة-1 وبالتالي ظهور الرسالة الآتية:



استبدال النصوص Replace

أحيانا قد نحتاج إلى استبدال نص معين بنص آخر، إذا أردنا عمل ذلك فإن لغة الفيجوال بيسك توفر لنا الدالة Replace التي تقوم بهذه العملية كما يبين المثال الآتي:

```
Dim FalseText As String = "الرياض عاصمة العراق"
Dim TrueText As String = FalseText.Replace("الرياض", "بغداد")
MsgBox(TrueText)
```

المتغير الأول يحتوي على عبارة خاطئة مفادها أن الرياض عاصمة بغداد، وقمنا بتصحيح العبارة من خلال الدالة Replace التي مكنتنا من استبدال مدينة الرياض بالمدينة الصحيحة التي تعد عاصمة للعراق.

المتغير TrueText سيحتوي على العبارة التالية: بغداد عاصمة العراق.

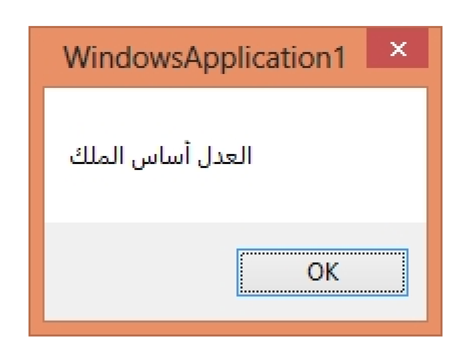
إدراج النصوص Insert

إذا أردنا أن نضيف نصا ما إلى نص آخر مع إمكانية تحديد الرتبة التي نريد إدخال النص الجديد فيها، فإن الدالة Insert تفي بالغرض وتسمح لنا بهذه العملية كما يوضح المثال الآتي:

```
Dim Expression As String = "أساس الملك"
Dim FinalExpression As String = Expression.Insert(0, "العدل  ")
```

```
        MsgBox(FinalExpression)
```

المتغير الأول يحتوي على عبارة ناقصة هي: أساس الملك، فقمنا بإضافة الكلمة الناقصة في الرتبة 0 وهي الكلمة: العدل، وبالتالي ستصبح قيمة المتغير الثاني FinalExpression هي:



مقارنة النصوص Compare

قد نحتاج في بعض الأحيان إلى إجراء مقارنات على النصوص من أجا التحقق من أن نصين متوافقان أو غير متوافقان، لعمل ذلك توجد عندنا الدالة Compare والدالة CompareTo اللتان تقومان بمقارنة نصين فإن تم إرجاع القيمة 0 فهذا يعني أن النصين متماثلان، وإن تم إرجاع القيمة 1 فهذا يعني أن القيمتين غير متوافقتين، كما يعرض لنا المثال التالي الذي استخدمنا فيه الدالتين معا:

```
        Dim First_String As String = "Khalid"

        Dim Second_String As String = "Hussain"

        Dim Result As Integer

        Result = String.Compare(First_String, Second_String)

        If Result = 0 Then
            MsgBox("النص الأول مماثل للنص الثاني")
```

```
        Else
            MsgBox("النص الأول غير مماثل للنص الثاني  ")
        End If
        'أو
        Result = First_String.CompareTo(Second_String)

        If Result = 0 Then
            MsgBox("النص الأول مماثل للنص الثاني  ")
        Else
            MsgBox("النص الأول غير مماثل للنص الثاني  ")
        End If
```

الدالة Format

قبل أن نتطرق لشرح كيفية التحكم في عرض العملات المالية والتواريخ وغيرهم، ينبغي أن نستعرض أولا الدالة Format التي تنتمي إلى الفئة String والتي تسمح لنا بالتعامل مع مختلف القيم، بحيث نحدد رتبة كل قيمة وشكل العرض بين مزدوجتين ثم بعد ذلك مورد اسم المتغير على شكل برامتر للدالة Format كما يبين المثال التالي:

```
        Dim First_Name As String = "Khalid"
        Dim Last_Name As String = "ESSAADANI"
        Dim FullName As String
        FullName = String.Format("My name is: {0} {1}",
First_Name, Last_Name)
        MsgBox(FullName)
```

لاحظ أننا حددنا داخل الدالة Format ترتيب المتغيرات وشكل الظهور، حيث تركنا فراغا بين الاسم وبين النسب، عند تنفيذ الكود أعلاه سوف نحصل على هذه النتيجة:

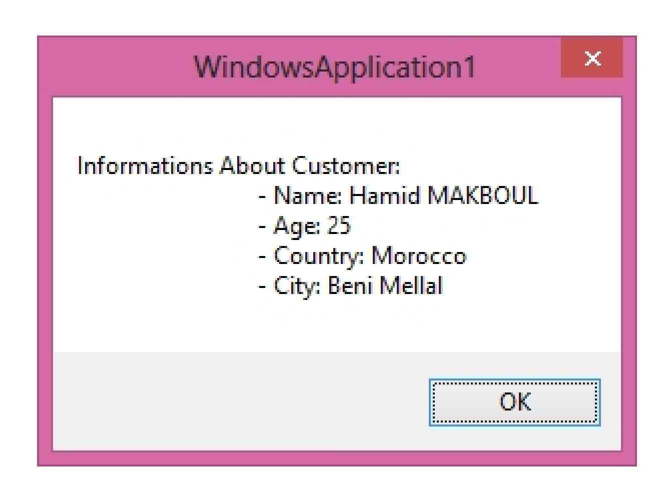


وهذا مثال آخر للاستئناس بالدالة Format:

```
Dim Name As String = "Hamid MAKBOUL"
Dim Age As Byte = 25
Dim Country As String = "Morocco"
Dim City As String = "Beni Mellal"

MsgBox(String.Format("Informations About Customer:
                        - Name: {0}
                        - Age: {1}
                        - Country: {2}
                        - City: {3}", Name, Age,
Country, City))
```

سيتم عرض معلومات الشخص بشكل منسق حسب الترتيب الذي حددناه في الدالة Format، إذا قمنا بتنفيذ الكود أعلاه سوف نحصل على الرسالة التالية:

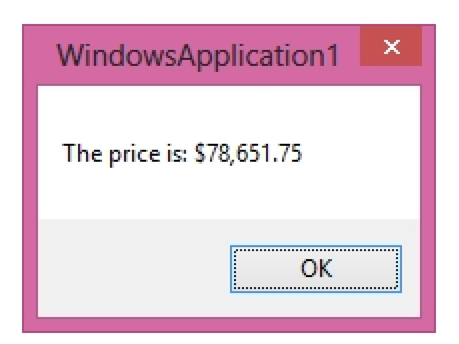


تنسيق العملات

لتنسيق العملات وعرضها بالشكل الصحيح، نستخدم الدالة Format ونضع في تنسيق القيمة المالية الرمز c دلالة على أن التنسيق سيكون على شكل عملة Currency كما يوضح الكود التالي:

```
Dim Price As Double = 78651.75
MsgBox(String.Format("The price is: {0:c}", Price))
```

سيتم تنسيق القيمة العشرية بالشكل المناسب لعرض المبالغ المالية في نظام التشغيل، عند تنفيذ الكود أعلاه سوف تحصل على نتيجة مماثلة لما يلي مع اختلاف في التنسيق والعملة حسب إعدادات نظام التشغيل عند:

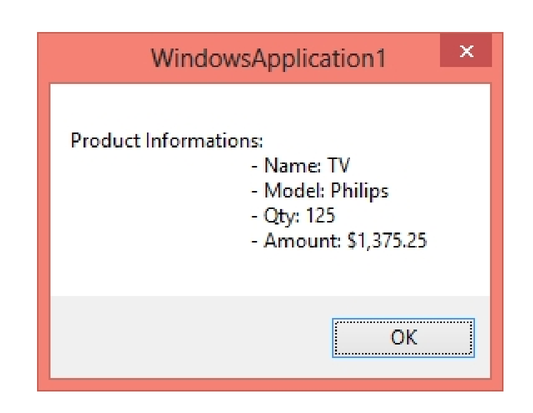


في المثال التالي سنقوم بنفس العمل، لكن مع استخدام متغيرات أخرى داخل الدالة Format:

```
Dim Name As String = "TV"
Dim Model As String = "Philips"
Dim Quantity As Short = 125
Dim Amount As Double = 1375.25
MsgBox(String.Format("Product Informations:
                          - Name: {0}
```

```
                                    - Model: {1}
                                    - Qty: {2}
                                    - Amount: {3:c}", Name,
Model, Quantity, Amount))
```

كل ما قمنا به في المثال أعلاه هو أننا أعلنا عن مجموعة من المتغيرات بما فيها المتغير العشري Amount الذي قمنا بتنسيق طريقة عرضه ليظهر بشكل مالي، عند تنفيذ هذه الشفرة سنحصل على نتيجة شبيهة بالصورة الآتية:



ويمكننا أيضا التحكم في شكل التنسيق، بحيث يمكننا عرض المبالغ بالشكل الذي نريد، كأن نعرض الوحدات والمئات والألوف وغيرهم مفصولين بفاصل آخر بدل الفاصل أو أن نقوم بوضع الأصفار في الأرقام الناقصة في كل جزء لتسهل قراءة المبالغ أو أن نحدد عدد الأرقام بعد الفاصلة العشرية، إذا أردنا عمل ذلك فما علينا سوى وضع التنسيق الذي نريد على شكل رموز # في الدالة Format كما يوضح المثال التالي:

```
Dim Amount As Double = 45612.658
MsgBox(String.Format("{0:### ###.##}", Amount))
```

في هذا المثال سيتم عرض الألوف مفصولة عن المئات بفراغ وسيتم عرض رقمين فقط وراء الفاصلة العشرية كما يلي:



يمكننا تغيير شكل التنسيق كنا نحب، مثلا نعرض بدل الفراغ فاصلة، ونظهر بعد الفاصلة العشرية رقما واحدا فقط:

```
Dim Amount As Double = 45612.658
MsgBox(String.Format("{0:###,###.#}", Amount))
```

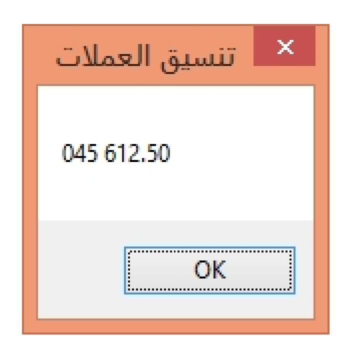
هذه المرة ستكون النتيجة هكذا:



إذا أحببنا أن نستعيض بأصفار عن الأماكن الناقصة في الأعداد والتي ليس لها تأثير حسابي لكنها تسهل عملية قراءة الأعداد يجب أن نضع بدل الرمز # القيمة 0 كما يوضح المثال التالي:

```
Dim Amount As Double = 45612.5
MsgBox(String.Format("{0:000 000.00}", Amount))
```

ستكون النتيجة كما يلي:



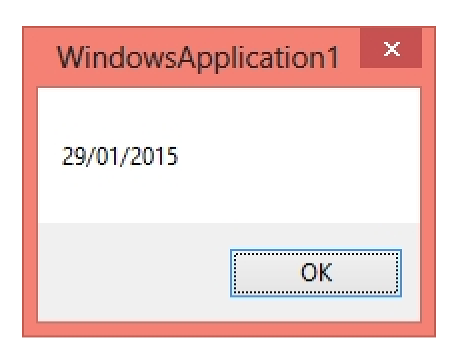
تنسيق التاريخ والوقت

بنفس الصيغة التي رأيناها في تنسيق العملات بواسطة الدالة Format يمكننا تنسيق القيم التاريخية لكن عبر تحديد شكل التنسيق بما يتلاءم مع طبيعة التاريخ والوقت الذي نود عرضه، افتراضيا نقوم بوضع الحرف d دلالة على Date من أجب تنسيق القيمة على شكل تاريخ.

في الفيجوال بيسك يوجد لدينا الخاصية ()Now التي تقوم بإرجاع التاريخ والوقت الحاليين في نظام التشغيل، وهي القيمة التي سنقوم بالعمل على تنسيقها في المثال التالي:

```
Dim CurrentDate As Date = Now
MsgBox(String.Format("{0:d}", CurrentDate))
```

في المثال فوقه، قمنا بالإعلان عن متغير اسمه CurrentDate يستقبل قيمة الخاصية Date، وبعد ذلك قمنا بعرض قيمته في رسالة بواسطة الدالة Format وحددنا في شكل التنسيق الرمز d للإشارة أن القيمة تاريخية، عند تنفيذ الطود سنحصل على النتيجة التالية:



يمكننا التحكم في تنسيق التاريخ بالشكل الذي نريد، بحيث يمكننا تغيير ترتيب الأيام والأشهر والسنوات والساعات والدقائق والثواني، سنورد في المثال التالي مجموعة من التنسيقات وبعدها سنفرد صورة لنتيجة كل تنسيق لكي نفهم معناه:

```
Dim CurrentDate As Date = Now

'عرض التاريخ كاملا
MsgBox(String.Format("{0:d}", CurrentDate))

'عرض اليوم و الشهر فقط
MsgBox(String.Format("{0:dd/MM}", CurrentDate))

'عرض اليوم فقط
MsgBox(String.Format("{0:dd}", CurrentDate))

'عرض الشهر و السنة فقط
MsgBox(String.Format("{0:MM/yyyy}", CurrentDate))
```

```
        'عرض السنة فقط
        MsgBox(String.Format("{0:yyyy}", CurrentDate))

        'عرض اليوم أولا، الشهر ثانيا، ثم السنة
        MsgBox(String.Format("{0:yyyy/MM/dd}", CurrentDate))

        'عرض التاريخ والوقت كاملين
        MsgBox(String.Format("{0:ss:mm:hh - yyyy/MM/dd}",
CurrentDate))

        'عرض الثواني فقط
        MsgBox(String.Format("{0:ss}", CurrentDate))

        'عرض الدقائق فقط
        MsgBox(String.Format("{0:mm}", CurrentDate))

        'عرض الساعات فقط
        MsgBox(String.Format("{0:hh}", CurrentDate))

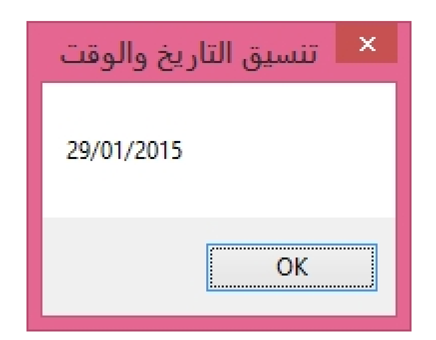
        'عرض الوقت كاملا
        MsgBox(String.Format("{0:ss:mm:hh}", CurrentDate))

        'عرض الدقائق والساعات
        MsgBox(String.Format("{0:mm:hh}", CurrentDate))
```

الأمر التالي:

```
        'عرض التاريخ كاملا
        MsgBox(String.Format("{0:d}", CurrentDate))
```

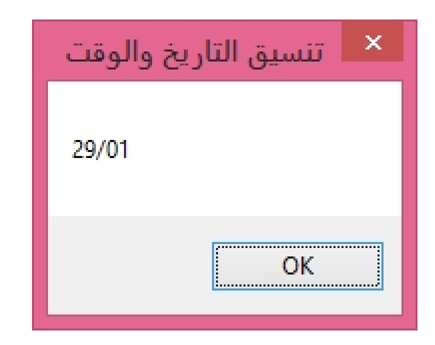
نتيجته:



الأمر التالي:

```
'عرض اليوم والشهر فقط
MsgBox(String.Format("{0:dd/MM}", CurrentDate))
```

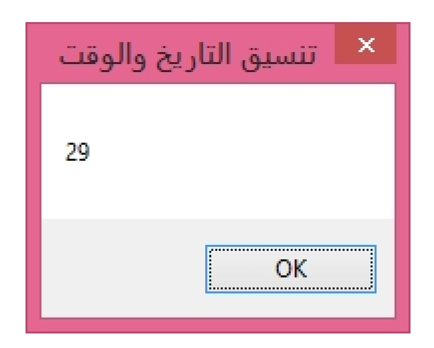
نتيجته:



الأمر التالي:

```
'عرض اليوم فقط
MsgBox(String.Format("{0:dd}", CurrentDate))
```

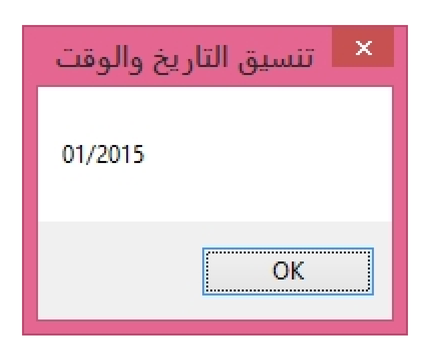
نتيجته:



الأمر التالي:

```
'عرض الشهر و السنة فقط
MsgBox(String.Format("{0:MM/yyyy}", CurrentDate))
```

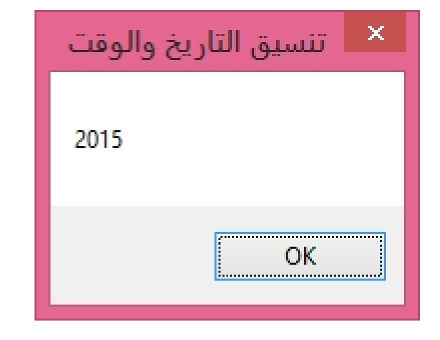
نتيجته:



الأمر التالي:

```
'عرض السنة فقط
MsgBox(String.Format("{0:yyyy}", CurrentDate))
```

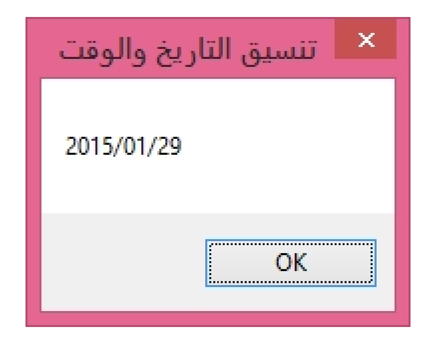
نتيجته:



الأمر التالي:

```
'عرض اليوم أولا، الشهر ثانيا، ثم السنة
MsgBox(String.Format("{0:yyyy/MM/dd}", CurrentDate))
```

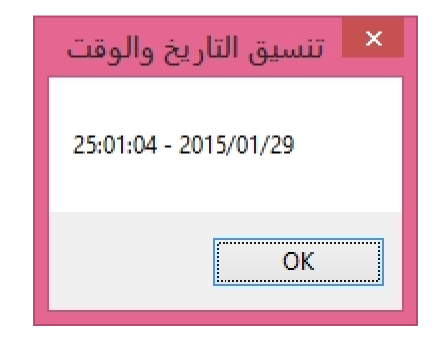
نتيجته:



الأمر التالي:

```
        'عرض التاريخ و الوقت كاملين
        MsgBox(String.Format("{0:ss:mm:hh - yyyy/MM/dd}", 
CurrentDate))
```

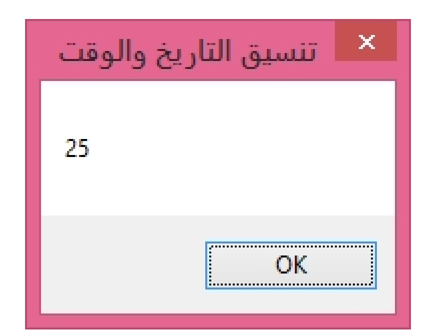
نتيجته:



الأمر التالي:

```
        'عرض الثواني فقط
        MsgBox(String.Format("{0:ss}", CurrentDate))
```

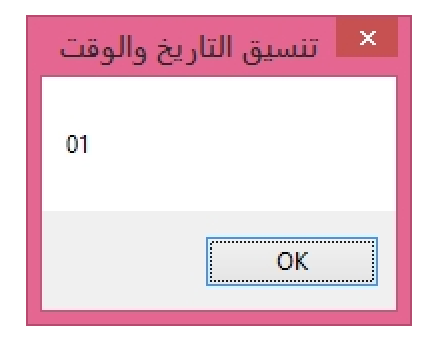
نتيجته:



الأمر التالي:

```
'عرض الدقائق فقط
MsgBox(String.Format("{0:mm}", CurrentDate))
```

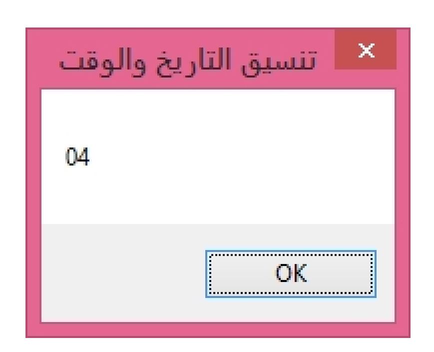
نتيجته:



الأمر التالي:

```
'عرض الساعات فقط
MsgBox(String.Format("{0:hh}", CurrentDate))
```

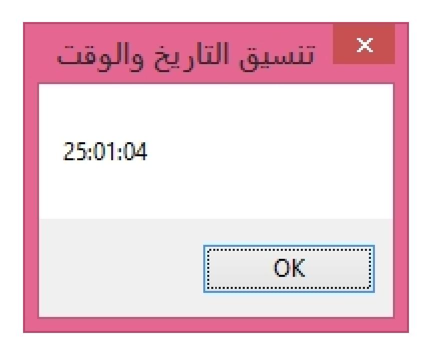
نتيجته:



الأمر التالي:

```
'عرض الوقت كاملا
MsgBox(String.Format("{0:ss:mm:hh}", CurrentDate))
```

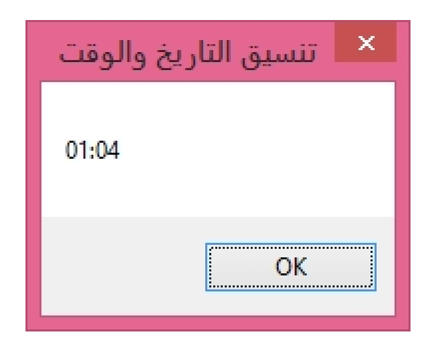
نتيجته:



الأمر التالي:

```
'عرض الدقائق و الساعات
MsgBox(String.Format("{0:mm:hh}", CurrentDate))
```

نتيجته:



الدوال والإجراءات Functions and Procedures

الدوال والإجراءات من المفاهيم البرمجية الأساسية المهمة والتي تمثل مجموعة من الأوامر البرمجية التي تؤدي عملا معينا، بحيث تكون هذه الأوامر مجمعة تحت اسم ما، وعند استدعاء هذا الاسم يتم تنفيذ الأوامر المنضوية داخله، والغرض من استخدام الدوال والإجراءات هو منع تكرار الأوامر في المشاريع البرمجية، مثلا لو عندي أوامر برمجية تؤدي عملا ما وسأحتاج إلى تكرارها في أماكن عديدة من البرنامج أقوم بتجميعها داخل إجراء Procedure أو دالة Function حسب العمل الذي تؤديه هذه الأوامر، ثم نقوم بالنداء على اسم الدالة أو الإجراء في كل مكان نحتاج فيه إلى تنفيذ هذه الأوامر.

الإجراءات هي مجموعة من الأوامر التي تؤدي عملا ما دون أن ترجع لنا قيمة كنتيجة لتنفيذ هذه الأوامر، وصيغة إنشاء الإجراءات في لغة الفيجوال بيسك تكون بواسطة الكلمة المحجوزة Sub ثم نورد اسم الإجراء ونضع بين القوسين البرامترات التي يحتاجها الإجراء كما يلي:

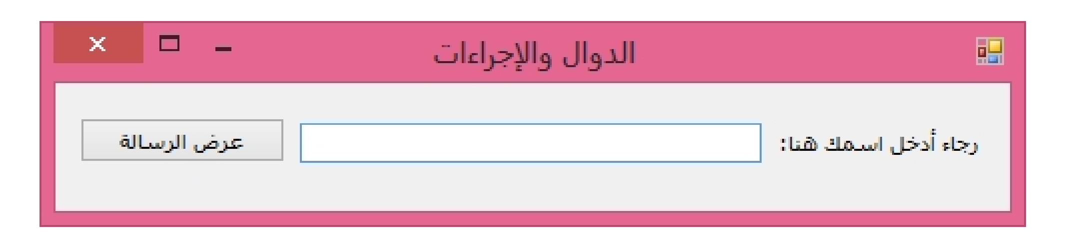
```
    Sub Procedure_Name(ByVal Param1 As Data_Type, ByRef
Param2 As Data,....)
        'do that
    End Sub
```

البرامترات هي متغيرات تنتظرها الإجراءات والدوال من أجل استخدامها للحصول على نتيجة معينة، ويكون الإعلان عن البرامترات بين قوسي الإجراء أو الدالة مع تحديد نوع بيانات كل برامتر وطريقة الحجز إما بالقيمة ByVal أو بالمرجع ByRef وسنوضح الفرق بين الطريقتين فيما بعد إن شاء اللّه.

بعد تحديد اسم الإجراء والبرامترات التي سيحتاجها (إذا لم يكن الإجراء في حاجة إلى برامترات نترك القوسين فارغين) نقوم بكتابة الأوامر التي على الإجراء تنفيذها عند كل استدعاء.

في المثال التالي سوف نتعرف على كيفية إنشاء إجراء يقوم بحساب استقبال اسم المستخدم وعرضه في رسالة.

قم بإنشاء مشروع جديد من نوع Windows Forms Application ثم ضع على الفورم الأدوات التالية:



الأداة	اسمها	دورها
Label	lblName	من أجل عرض النص
TextBox	txtName	من أجل إدخال الاسم
Button	btnShow	من أجل عرض الرسالة الترحيبية

بعد ذلك ادخل إلى الكود وقم بإنشاء الإجراء التالي أسفل الجزء Public Class Form1 كما يلي:

```
    Sub ShowMessage(ByVal Name As String)
        MsgBox(String.Concat(" السلام عليكم ورحمة الله وبركاته سيد: ",
Name))
    End Sub
```

الإجراء يحمل الاسم ShowMessage ويستقبل البرامتر النصي Name، بعد ذلك يقوم الإجراء بعرض رسالة ترحيبية تدمج بين عبارة: السلام عليكم ورحمة الله وبركاته سيد والاسم الممرفي البرامتر.

لاستدعاء هذا الإجراء نأتي إلى المكان الذي نريد تنفيذه فيه وهو الحدث Click الخاص بالزر btnShow ثم نكتب اسم الإجراء ونفتح القوس فيطلب منا كتابة قيمة نصية كما يلي:

```
WindowsApplication1    Form1.vb*    Form1.vb [Design]*
WindowsApplication1    btnShow    Click
Public Class Form1
    Sub ShowMessage(ByVal Name As String)
        MsgBox(String.Concat("السلام عليكم ورحمة الله وبركاته سيد : ", Name))
    End Sub

    Private Sub btnShow_Click(sender As Object, e As EventArgs) Handles btnShow.Cl
        ShowMessage()
    End  Form1.ShowMessage(Name As String)
```

سنضع في البرامتر النصي الذي ينتظره الإجراء ShowMessage القيمة النصية المدخلة في مربع النص txtName كما يلي:

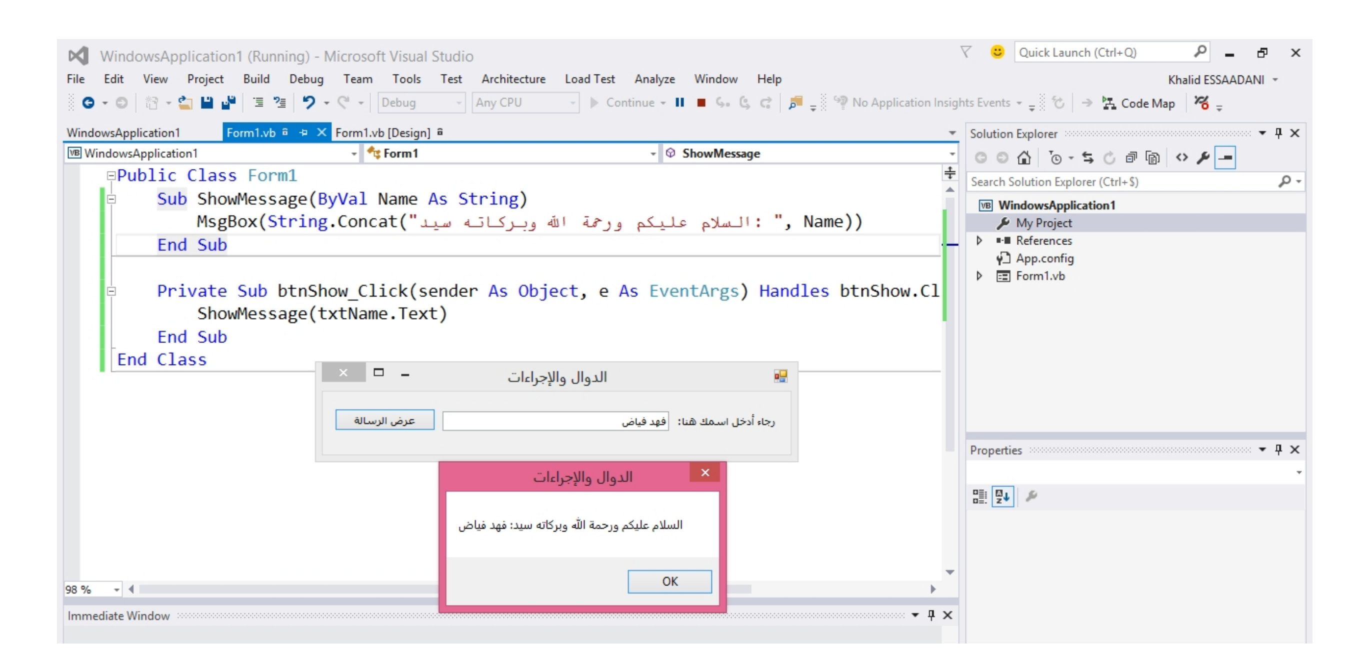
```
    Private Sub btnShow_Click(sender As Object, e As
EventArgs) Handles btnShow.Click

        ShowMessage(txtName.Text)

    End Sub
```

الآن سيتم استبدال البرامتر Name الذي ينتظره الإجراء ShowMessage بالقيمة النصية التي سيدخلها المستخدم في مربع النص وسيتم عرض هذه القيمة النصية مدموجة

برسالة الترحيب، وبالتالي عند تنفيذ الكود والنقر على الزر btnShow سيتم عرض الرسالة بالشكل الذي حددناه في الإجراء كما يلي:



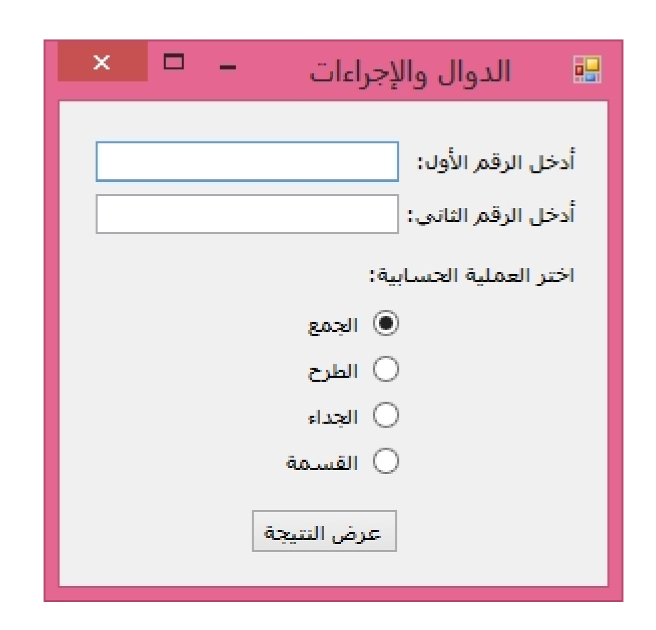
وهنا الكود كاملا:

```
Public Class Form1
    Sub ShowMessage(ByVal Name As String)
        MsgBox(String.Concat(" السلام عليكم ورحمة الله وبركاته سيد: ",
Name))
    End Sub

    Private Sub btnShow_Click(sender As Object, e As
EventArgs) Handles btnShow.Click
        ShowMessage(txtName.Text)
    End Sub
End Class
```

الدوال Functions شبيهة جدا بمفهوم الإجراءات إلا أنها تتميز عن هذه الأخيرة في كونها تسمح لنا بالحصول على النتيجة التي يؤديها تنفيذ أوامر الدالة، ويكون الإعلان عن الدالة بالكلمة Function مع تحديد نوع القيمة التي نود إرجاعها بعد تنفيذ الأوامر، كما سنرى في المثال التالي الذي يقوم بالعمليات الحسابية.

قم بإنشاء مشروع من نوع Windows Forms Application وضع على الفورم الأدوات التالية:



الأداة	اسمها	دورها
Label	lblNumber1	من أجل عرض النص الخاص بالرقم الأول
Label	lblNumber2	من أجل عرض النص الخاص بالرقم الثاني
Label	lblChoose	من أجل عرض النص الخاص باختيار العملية الحسابية
TextBox	txtNumber1	من أجل إدخال الرقم الأول
TextBox	txtNumber2	من أجل إدخال الرقم الثاني
RadiButton	rbAdd	من أجل اختيار عملية الجمع
RadiButton	rbSubs	من أجل اختيار عملية الطرح
RadiButton	rbMul	من أجل اختيار عملية الجداء

RadiButton	rbDiv	من أجل اختيار عملية القيمة
Button	btnCalculate	من أجل تنفيذ عملية الحساب

بعد أن تقوم بتصميم الفورم وتغيير أسماء الأدوات وتعديل خصائصها، قم بالولوج إلى محرر الكود وأنشئ الدالة التالية التي تقوم بحساب العددين المدخلين بناء على نوع العملية الذي تم اختياره وترجع لنا النتيجة الحاصلة كما يلي:

```
    Function Calculate(ByVal Number1 As Double, ByVal Number2 
As Double) As Double

        Dim Result As Double

        If rbAdd.Checked = True Then
            Result = Number1 + Number2
        ElseIf rbSubs.Checked = True
            Result = Number1 - Number2
        ElseIf rbMul.Checked = True
            Result = Number1 * Number2
        ElseIf rbDiv.Checked = True
            Result = Number1 / Number2
        End If
        Return Result

    End Function
```

قمنا بالإعلان عن الدالة Calculate من خلال الكلمة المحجوزة Function وقمنا بتحديد النوع الذي سوف تكون عليه نتيجة الدالة وهو Double وقبل ذلك حددنا البرامترات التي ستحتاجها الدالة من أجل إجراء الأوامر التي تؤدي إلى الحصول على النتيجة المتوخاة.

بعد ذلك قمنا داخل الدالة بالإعلان عن المتغير Result من نوع عشري Double الذي يأخذ قيمته بناء على نوع العملية الذي يتم اختياره، فإن تم احتيار الجمع كانت

قيمته هي مجموع المتغيرين Number1 و Number2 وهكذا دواليك مع باقي العمليات الحسابية.

في آخر المطاف نقوم بإرجاع القيمة Result التي ستجتوي على نتيحة العملية الحسابية التي تم تحديدها، هذه النتيجة يمكننا استخدامها كما نشاء خارج مجال الدالة.

الآن سوف نأتي إلى الحدثِ Click الخاص بالزر btnCalculate ونكتب أمر استدعاء الدالة Calculate مع تمرير قيم البرامترات Number1 و Number2 اللذين سيأخذان القيم بدورهما من المربعين النصيين اللذين وضعناهما على الفورم:

```
    Private Sub btnCalcul_Click(sender As Object, e As
EventArgs) Handles btnCalcul.Click

        Dim N1, N2, Result As Double

        N1 = Val(txtNumber1.Text)
        N2 = Val(txtNumber2.Text)

        Result = Calculate(N1, N2)

        MsgBox(String.Format("The Result Is: {0}", Result))
    End Sub
```

لاحظ أننا أعلنا عن ثلاثة متغيرات عشرية، الأول N1 سيستقبل القيمة المدخلة في مربع النص txtNumber1 بعد أن يحولها إلى قيمة رقمية، والمتغير الثاني N2 خاص ب txtNumber2 والمتغير الثالث Result سيحتوي على النتيجة الحسابية التي ترجعها الدالة Calcuate.

ثم بعد ذلك أظهرنا النتيجة الحاصلة في رسالة، ويمكننا اختزال الأوامر أعلاه كاملة في سطر واحد كما يلي:

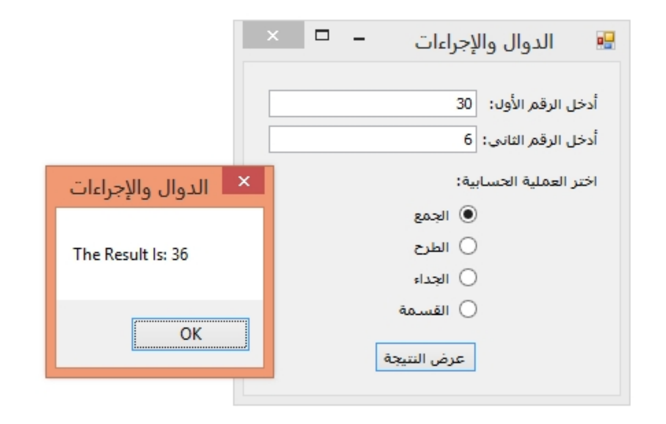
```
    Private Sub btnCalcul_Click(sender As Object, e As EventArgs) Handles btnCalcul.Click

        MsgBox(String.Format("The Result Is: {0}", Calculate(Val(txtNumber1.Text), Val(txtNumber2.Text))))

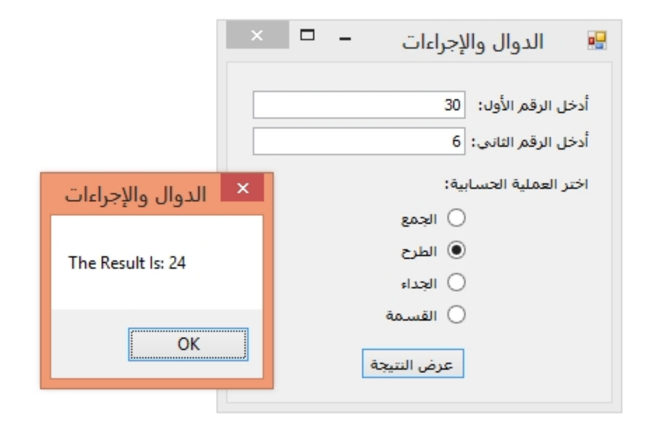
    End Sub
```

عند تنفيذ الكود أعلاه وإدخال أية قيم رقمية في مربعات النصوص سوف نحصل على نتائج مماثلة لما يلي:

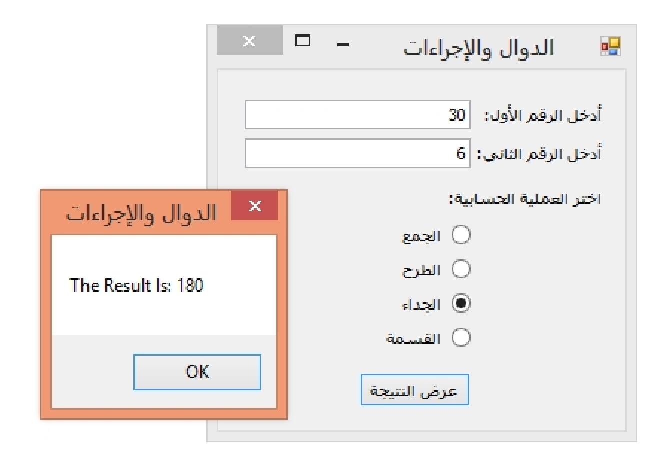
نتيجة الجمع:



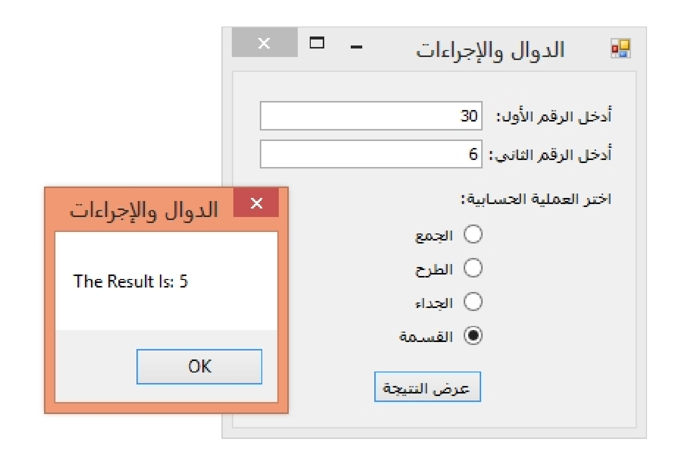
نتيجة الطرح:



نتيجة الجداء:



نتيجة القسمة:



تمرير البرامترات

ذكرنا أثناء حديثنا عن الدوال والإجراءات أنه يمكننا تمرير البرامترات من خلال القيمة ByVal ومن خلال المرجع ByRef والآن بحول الله سوف نتعرف على الفرق بين الصيغتين.

عند تمرير برامتر بالقيمة By Value فإن قيمة هذا البرامتر تتغير داخل الدالة فقط، وقيمة المتغير الأصلي الذي نضعه مكان البرامترات عند الاستدعاء لا تتغير، لأن الدالة أو الإجراء يقوم بأخذ نسخة Copy من قيمة المتغير الخارجي ويضعها مكان البرامتر دون أن يمس القيمة الأصلية للمتغير.

بينما تمرير القيمة بالمرجع By Reference يقوم بإلغاء القيمة الأصلية للمتغير الخارجي ويستبدلها بالقيمة الناتجة عن أوامر الإجراء أو الدالة.

لكي نفهم الاختلاف بين تمرير البرامترات بالقيمة وبين تمرير البرامترات بالمرجع بشكل أوضح، سنقوم بإنشاء نفس الإجراء لكن في كل مرة سنغير طريقة التمرير لنقارن النتائج ومن ثم نستنبط الفروق.

ادخل إلى محرر الكود وأنشئ الإجراء التالي الذي يمرر قيمة البرامتر Number بالقيمة و يحسب جداءها في 2 ويعرض النتيجة في رسالة:

```
Sub CalcByVal(ByVal Number As Integer)
    Number *= 2
    MsgBox("بالقيمة: " & Number)
End Sub
```

لو مررنا للبرامتر Number القيمة 25 سيتم ضربها في 2 وتظهر النتيجة 50 في رسالة.

الآن قم بإنشاء نفس الإجراء باسم مختلف لكن هذه المرة نوع التمرير سيكون بالمرجع By Referenece كما يلي:

```
Sub CalcByRef(ByRef Number As Integer)
    Number *= 2
    MsgBox("بالمرجع: " & Number)
End Sub
```

الإجراء أعلاه يؤدي نفس النتيجة التي يؤديها الإجراء الأول، لكن تعال بنا نختبرهما معا لنخلص إلى الفرق الجوهري بين القيمة وبين المرجع.

ادخل إلى أي مكان تريد تنفيذ الإجراءين فيه، وليكن مثلا الحدث Click لزر ما، واكتب الكود التالي:

```
Dim Number As Integer = 16

'قيمة المتغير قبل النداء
MsgBox("قبل النداء: " & Number)

'قيمة المتغير أثناء النداء على إجراء يمرر بالقيمة
CalcByVal(Number)

'قيمة المتغير بعد النداء على إجراء يمرر بالقيمة
MsgBox("القيمة بعد النداء على إجراء يمرر بالقيمة " & Number)

'قيمة المتغير أثناء النداء على إجراء يمرر بالمرجع
CalcByRef(Number)

'قيمة المتغير بعد النداء على إجراء يمرر بالمرجع
MsgBox("بعد النداء على إجراء يمرر بالمرجع: " & Number)
```

الآن سنقوم بشرح الكود أعلاه سطرا سطرا لنفهم مالذي يجري.

السطر الأول:

```
Dim Number As Integer = 16
```

في هذا السطر أعلنا عن متغير رقمي Number قيمته البدئية هي 16، بمعنى أنه تم حجز خانة في الذاكرة ووضعت فيها القيمة الرقمية 16.

السطر الثاني:

```
'قيمة المتغير قبل النداء
MsgBox("قبل النداء:  " & Number)
```

هنا قمنا بعرض قيمة المتغير قبل أن نستخدمها في أية إجراء، أي أن النتيجة يتكون هي 16.

السطر الثالث:

```
'قيمة المتغير أثناء النداء على إجراء يمرر بالقيمة
CalcByVal(Number)
```

هنا قمنا باستدعاء الإجراء CalcByVal الذي سيأخذ نسخة من قيمة المتغير Number وهي 16 وسيدخلها في العملية الحسابية الجدائية دون أن يمس قيمة المتغير الأصلية، يعني أن قيمة المتغير Number ستبقى 16 بينما داخل الدالة ستصبح هي 2 * 16 أي 32.

السطر الرابع:

```
'قيمة المتغير بعد النداء على إجراء يمرر بالقيمة
MsgBox("القيمة بعد النداء على إجراء يمرر بالقيمة " & Number)
```

في هذا السطر ستظهر 16 لأن قيمة المتغير الأصلية لم تتغير لأن الإجراء السابق استقبل قام بالتمرير بالقيمة أي أنه أخذ نسخة فقط واستخدمها في العملية الحسابية دون أن يمس قيمة المتغير الأصلية، بمعنى أن الرسالة ستظهر 16.

السطر الخامس:

```
'قيمة المتغير أثناء النداء على إجراء يمرر بالمرجع
CalcByRef(Number)
```

الآن قمنا باستدعاء الإجراء الذي يمرر القيمة بالمرجع، وبالتالي سيتم إلغاء قيمة المتغير Number وسيتم استبدالها بالقيمة الجديدة الناتجة عن الإجراء، أي أن الإجراء CalcByRef سيظهر القيمة 2 * 16 أي 32 وسيتم إلغاء القيمة 16 من المتغير Number وسيتم استبدالها ب 32.

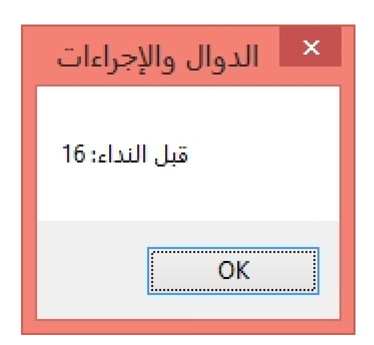
السطر السادس:

```
'قيمة المتغير بعد النداء على إجراء يمرر بالمرجع
MsgBox("بعد النداء على إجراء يمرر بالمرجع: " & Number)
```

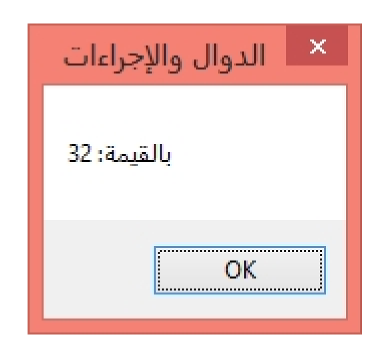
ستظهر الرسالة أعلاه القيمة 32 لأن التمرير قبل هذه الرسالة كان بالمرجع وبالتالي سيتم إلغاء القيمة الأصلية 16 واستبدالها ب 32.

فيما يلي النتائج الظاهرة عند تنفيذ الكود أعلاه بالترتيب:

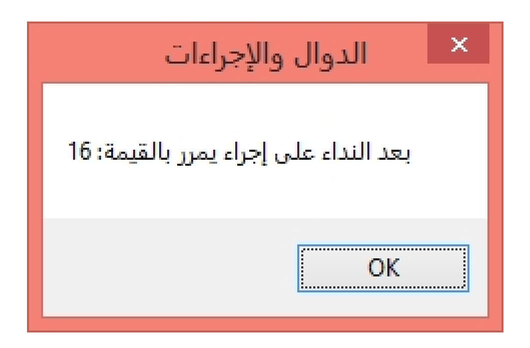
قيمة المتغير الأصلية:



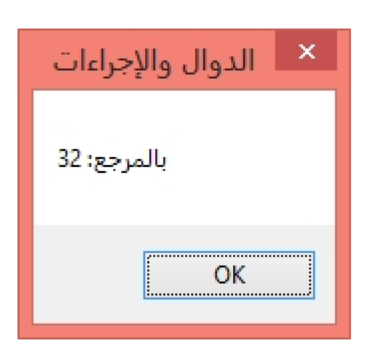
قيمة المتغير داخل الإجراء عند التمرير بالقيمة By Value:



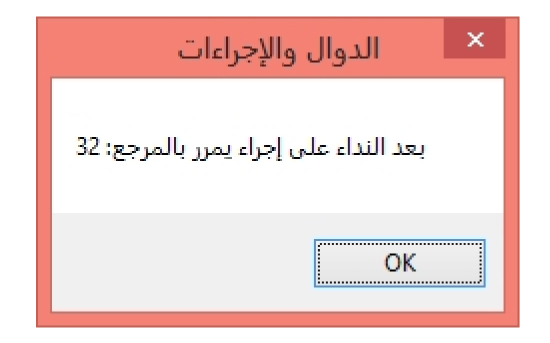
قيمة المتغير بعد الخروج من الإجراء:



قيمة المتغير عند التمرير بالمرجع By Reference:



قيمة المتغير بعد الخروج من الإجراء:



معالجة الاستثناءات Exceptions Handling:

الاستثناءات هي أخطاء تحدث عند اشتغال البرنامج أي في مرحلة التنفيذ Runtime، ويكون ذلك لأسباب عديدة، من بينها إدخال قيمة غير مناسبة، كمحاولة تخزين قيمة نصية في متغير رقمي، أو محاولة القيام بعملية القسمة على صفر، أو محاولة حذف ملف غير موجود أساسا، وغير ذلك...

لكي نتعرف على بعض الاستثناءات الممكن حدوثها، قم بإنشاء مشروع جديد من نوع Windows Forms Application وضع على الفورم الأدوات التالية:



الأداة	اسمها	دورها
Label	lblAge	من أجل عرض النص الخاص بالسن
TextBox	txtAge	من أجل إدخال السن
Button	btnShow	من أجل حساب ضعف السن

البرنامج أعلاه يطلب من المستخدم أن يدخل عمره ثم يقوم بعرض ضعق العمر (العمر * 2)، والمفروض أن يكون العمر على شكل قيمة رقمية، أي أن الكود الخاص بالزر btnShow ينبغي أن يكون على الشكل التالي:

```
    Private Sub btnShow_Click(sender As Object, e As EventArgs) Handles btnShow.Click

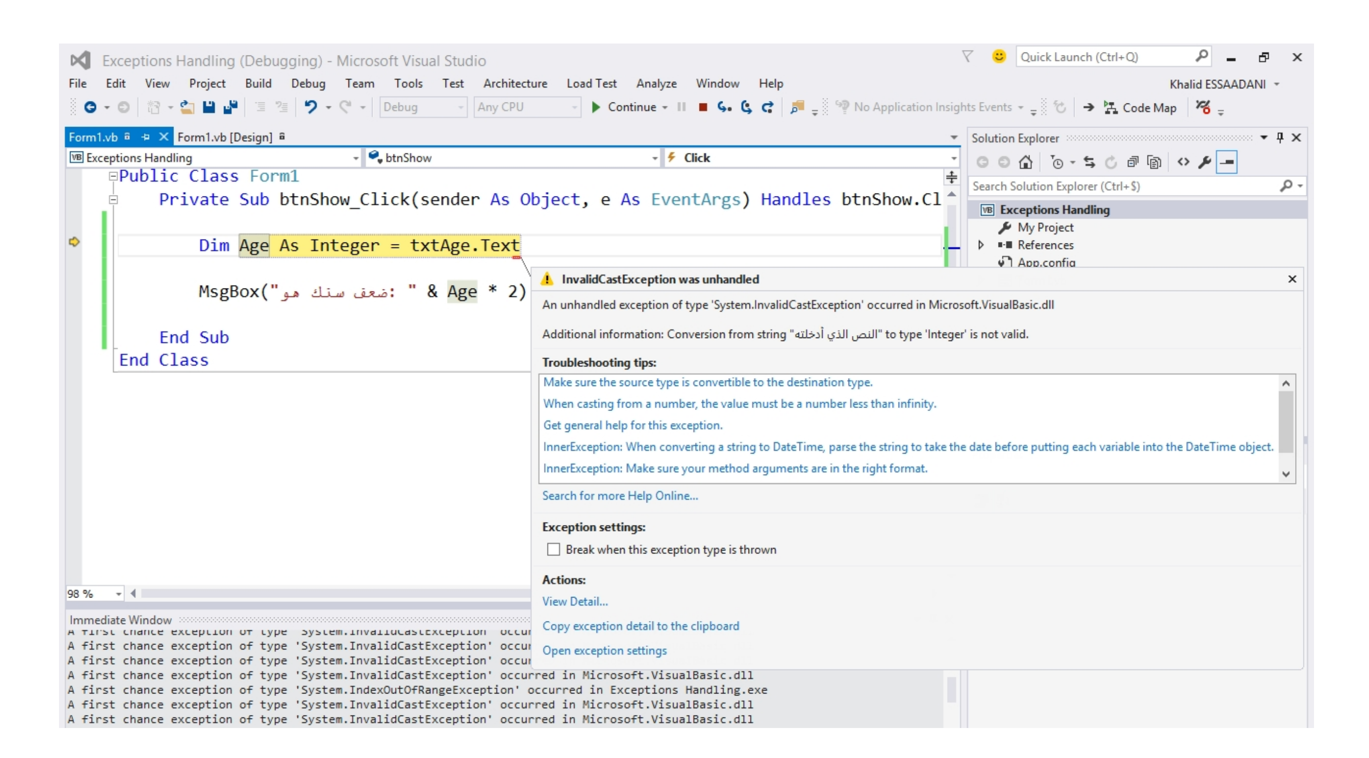
        Dim Age As Integer = txtAge.Text

        MsgBox("ضعف سنك هو: " & Age * 2)

    End Sub
```

الآن لو نفذنا البرنامج وأدخلنا قيمة رقمية في العمر سيتم حسابها على أحسن وجه، لكن ماذا لو قمنا بإدخال قيمة نصية؟

المتغير Age ينتظر قيمة رقمية، فإن نحن أسندنا إليه قيمة نصية سيحصل استثناء في البرنامج مفاده أن القيمة غير صحيحة Invalid Cast كما توضح الصورة الآتية التي تبين تفاصيل الاستثناء الحاصل:



رسالة الخطأ تفيد بأن استثناء من نوع Invalid Cast Exception قد حصل، أي أن القيمة المسندة للمتغير لا تتوافق مع نوع بياناته.

وأنت تبرمج ستقع في مثل هاته المشاكل باستمرار، لذلك يتوجب عليك معالجتها لكي يبقى برنامجك شغالا دون أن تتسبب مثل هاته الاستثناءات في إيقافه.

معالجة الاستثناءات المنظمة Structured Exception Handling

لتفادي حدوث هذه الاستثناءات وغيرها، توفر لنا لغة الفيجوال بيسك بعض آليات إدارة الاستثناءات أبرزها البنية Try...Catch...Finally وصيغتها كما يلي:

```
Try
    'الكود المراد معالجته
Catch ex As Exception
    'في حل وقوع استثناء في الكود سينفذ هذا الجزء
Finally
    'هذا الجزء سينفذ في جميع الحالات سواء حصل استثناء أم لم يحصل
End Try
```

نضع بعد الكلمة Try الكود الذي نريد التحقق من سلامته وخلوه من الاستثناءات والمشاكل، فإذا حصل خطأ ما سيتم الانتقال إلى الجزء الموجود بعد الكلمة Catch ليتم تنفيذه، غالبا في هذا الجوء نعرض رسالة خطأ لتنبيه المستخدم إلى المشكل الحاصل.

أما الجزء Finally وهو أمر اختياري لسنا ملزمين بكتابته: فنضع بعده الكود الذي نريده أن ينفذ في جميع الحالات سواء حصل خطأ في البرنامج أم لم يحصل، على سبيل المثال لو الكود يقوم بالاتصال بقاعدة بيانات فعلينا قطع الاتصال معها في نهاية الكود أي أن قطع الاتصال ينبغي أن يتم في جميع الحالات.

لكي نفهم الآلية Try...Catch بشكل أوضح، نعود إلى نفس المثال السابق الخاص بإدخال العمر، سنضع الكود الذي يتسبب في الاستثناء بعد Try ثم في Catch نضع ما نود القيام به:

```
Try
    Dim Age As Integer = txtAge.Text
    MsgBox("ضعف سنك هو: " & Age * 2)
Catch ex As Exception
    MsgBox("رجاء لقد حصل خطأ في البرنامج")
    txtAge.Clear()
    txtAge.Focus()
Finally
    MsgBox("هذا الجزء سينفذ في جميع الحالات")
End Try
```

الآن تعال بنا نفصل الكود جزء بجزء:

```
Try
```

```
Dim Age As Integer = txtAge.Text
MsgBox("ضعف سنك هو:  " & Age * 2)
```

في هذا الجزء وضعنا الكود الذي نود التحقق من سلامته ومن خلوه من الاستثناءات، فإن حصل خطأ يتم الانتقال إلى الجزء التالي:

```
Catch ex As Exception
    MsgBox("رجاء لقد حصل خطأ في البرنامج")
    txtAge.Clear()
    txtAge.Focus()
```

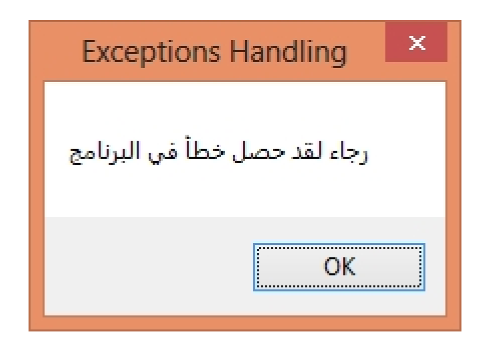
في هذا الجزء تم الإعلان عن متغير ex منو نوع Exception أي النوع الأصلي لكل الاستثناءات ويمكننا التخصيص وتحديد نوع الاستثناء إن كنا نعلمه أو نتركه هكذا إن كنا نجهله، بالنسبة لنوع الاستثناء المناسب في هذه الحالة فهو InvalidCastException بدل Exception، بعد ذلك قمنا بعرض رسالة خطأ، وقمنا بتفريغ مربع النص الخاص بإدخال السن ووضعنا المؤشر فيه من خلال الإجراء Focus.

بعدها مباشرة انتقلنا إلى الجزء الآتي:

```
Finally
    MsgBox("هذا الجزء سينفذ في جميع الحالات")
End Try
```

هذا الجزء سينفذ في جميع الحالات سواء حصل خطأ أو لم يحصل.

الآن لو نفذنا البرنامج وأدخلنا قيمة غير صحيحة مكان العمر، فلن يتوقف البرنامج بل سيعرض لنا رسالة الخطأ التالية:



ثم تظهر مباشرة بعدها الرسالة الموجودة في الجزء Finally:



يمكننا كتابة العديد من Catch حسب نوع الاستثناء الذي نود أن نعالجه، وسيتم تنفيذ الجزء المناسب في الوقت المناسب.

معالجة الاستثناءات غير المنظمة Unstructured Exception Handling:

يمكننا استخدام هذا النوع من الآليات أيضا من أجل معالجة الاستثناءات، وصيغتها تكون باستخدام الكلمة On Error كما توضح الأمثلة التالية:

```
        'عند الخطأ اذهب إلى الأمر
        On Error GoTo Label

Label: 'سيتم تنفيذ هذا الجزء عند الخطأ

        'عند الخطأ قم بتجاوزه
        On Error Resume Next
```

الآلية الأولى التي تكتب بالشكل التالي:

```
        'عند الخطأ اذهب إلى الأمر
        On Error GoTo Label
```

تعني أنه عند حدوث خطأ ما اذهب إلى الكود الذي يسبقه الاسم Label وقم بتنفيذه، مثلا لو طبقنا هذه الآلية على نفس المثال الذي نشتغل عليه والذي يقوم بحساب ضعف العمر، فإن الكود سيكتب بالشكل التالي:

```
        On Error GoTo ErrorMessage

        Dim Age As Integer = txtAge.Text

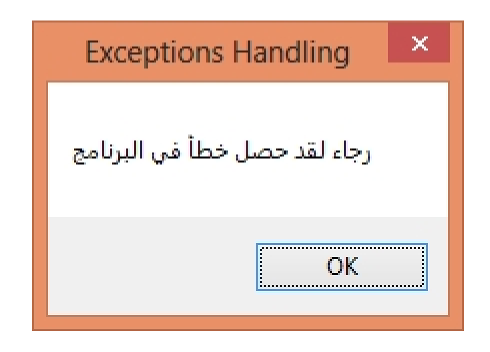
        MsgBox("ضعف سنك هو: " & Age * 2)

ErrorMessage:

        MsgBox("رجاء لقد حصل خطأ في البرنامج")
```

معنى هذا الكود هو: إذا وقع خطأ On Error في الكود فاذهب إلى المنطقة التي اسمها ErrorMessage وقم بتنفيذها.

عند تنفيذ هذا البرنامج سيتم عرض الرسالة التالية في حال حصول خطأ:



بالنسبة للكلمة ErrorMessage فيمكننا استبدالها بأية كلمة نريد، وتسمى برمجيا ب GoTo Label ودورها هو تحديد مكان لكود معين من أجل الانتقال إليه عند استخدام الأمر GoTo.

يوجد كذلك الآلية On Error Resume Next والتي تعني أن على البرنامج أن يتجاوز مكان الخطأ في حال حدوثه، كما يعرض معنا المثال التالي:

```
On Error Resume Next
Dim Age As Integer = txtAge.Text
MsgBox("ضعف سنك هو: " & Age * 2)
```

أثناء حصول خطأ في الكود أعلاه سيتم التغاضي عنه والانتقال إلى ما بعده لتنفيذه دون أن يتوقف البرنامج في مكان الخطأ.

إذا نفذنا الكود أعلاه سوف تظهر الرسالة التالية:



الخاتمة

تم بفضل الله وعونه الانتهاء من الجزء الأول من سلسلة الإكليل المكتوبة التابعة للحقيبة البرمجية للغة الفيجوال بيسك 2015، بعد أن استعرضنا جميع أساسيات البرمجة ومختلف المفاهيم التقنية بشرح ميسر ونماذج تطبيقية لتكريس وترسيخ المعلومات في الأذهان.

في الجزء الثاني من سلسلة الإكليل، سنتطرق إلى نمط البرمجة الكائنية التوجه Object Oriented Programming من خلال تعريف أبرز مفاهيمهما بنفس المنهج والبيداغوجيا التي اتبعناها في هذا الكتاب.

لكل شيء إذا ما تم نقصان، فإن وجدتم في طيات هذا الكتاب أخطاء لغوية أو تقنية أو لديكم ملاحظات واقتراحات لتحسين السلسلة فلا تترددوا بمراسلتنا عبر العناوين الالكترونية التالية:

mobarmijoun@gmail.com

how2progspace@gmail.com

وكذلك زيارتنا على موقع أكاديمية المبرمجين العرب:

www.mobarmjoun.com

ومتابعتنا عبر قناتنا على اليوتيوب وصفحتنا على الفيسبوك:

www.youtube.com/EssaadaniTV

www.facebook.com/EssaadaniPage